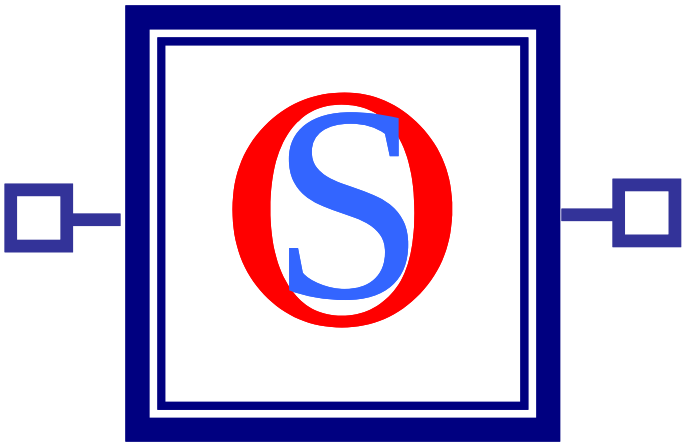
Universidade de Aveiro

Instituto de Telecomunicações

OSIP

Optical Simulator Platform

Manual do Programador



Versão 2.5

Julho 2006

Miguel Pereira nº 23799
Bruno Santos nº 23045

INDÍCE

Capítulo 1

Introdução

1.1 O que é o OSIP

O simulador OSIP – *Optical Simulation Platform* – foi pensado para simulações de sistemas de comunicações ópticas baseando-se na linguagem de programação proprietária existente no Programa MATLAB®.

É um simulador criado a partir de um conjunto de *m-files* e *functions*, que executadas de forma contínua, permitem a geração de uma interface gráfica de simulação baseada em janelas de fácil utilização para o utilizador.

O OSIP executa simulações sequenciais, ou seja, cada *m-file* de cada componente é executada uma vez, desempenhando em cada simulação as funções de um dado componente e obtendo-se resultados consoante os parâmetros introduzidos pelo

utilizador. Tem uma interface gráfica *user-friendly* e funcionalidades *drag-and-drop.* Permite realizar importações e exportações de componentes, agrupar componentes e até aceder a operações directamente do esquemático, através de ferramentas gráficas de utilização fácil e intuitiva.

Todo o processo de integração e configuração do simulador é gráfico não sendo necessário o uso de linha de comandos do Matlab para efectuar qualquer tipo de acção. Esta funcionalidade é realizada com a ajuda do GUIDE do MATLAB®, que é uma óptima ferramenta bastante prática e útil de construção de interfaces gráficas.

1.2 Motivação

Dadas as características de rápida evolução e crescente aplicação no mercado das telecomunicações e outros dos sistemas de comunicação ópticos, uma crescente necessidade de obter rapidamente resultados prévios sobre a aptidão ou não de determinada solução a determinado panorama.

Uma vez que as comunicações ópticas usam normalmente tecnologia de ponta, e assim bastante dispendiosa, é necessário obter, com alguma flexibilidade e a baixo custo, testes que permitam tomadas de decisão acertadas e em tempo útil. A forma mais utilizada para este fim é, normalmente, o uso de simuladores. Por estas razões estas ferramentas são de grande importância para investigadores e empresas de telecomunicações.

Embora existam no mercado este tipo de simuladores como é o caso do VPI®, estes são normalmente muito fechados em termos de possibilidade de inclusão ou alteração dos modelos utilizados.

Com este projecto pretendeu-se criar uma ferramenta aberta para este fim e que possa correr sobre uma plataforma que seja de corrente utilização (Matlab®).

1.3 Uso do Matlab: Linguagem de Programação versátil

MATLAB® é uma linguagem de programação apropriada ao desenvolvimento de aplicativos de natureza técnica. Como o próprio nome sugere, o MATLAB® é bem adequado àqueles que desejam implementar e testar soluções com facilidade e precisão (como num laboratório!), sem perder tempo com detalhes específicos de linguagem de programação. Para isso, possui facilidades de computação, visualização e programação, dentro de um ambiente amigável e de fácil aprendizagem.

O nome MATLAB® vem de *Matrix Laboratory*. MATLAB® foi originalmente desenvolvido para prover um acesso amigável ao tratamento de vectores e matrizes. Os elementos básicos da linguagem são exactamente os vectores e matrizes. Por esse motivo é importante que esses elementos e suas operações sejam bem entendidos para se obter o melhor do MATLAB®.

Actualmente o MATLAB® dispõe de uma biblioteca bastante abrangente de funções matemáticas, geração de gráficos e manipulação de dados que auxiliam muito o trabalho do programador. Possui ainda uma vasta colecção de bibliotecas - denominadas *toolboxes*- para áreas específicas como: estatística, processamento de imagens, processamento de sinais, finanças, telecomunicações, etc.

A linguagem e o ambiente de programação MATLAB® permitem ainda que o utilizador possa escrever as suas próprias bibliotecas em MATLAB®. Assim, o utilizador pode enriquecer a linguagem, incorporando a ela novas funções.

Possui ainda uma ferramenta de criação de objectos gráficos baseados em funções programáveis. Esta ferramenta – *GUIDE – Graphical User Interface Development Environment*, incorpora um conjunto de ferramentas para criar interfaces gráficas de utilizador (GUIs). Estas ferramentas simplificam largamente o processo de projectar e construir Interfaces Gráficas.

A figura seguinte apresenta a janela de interface que permite construir os GUIs.

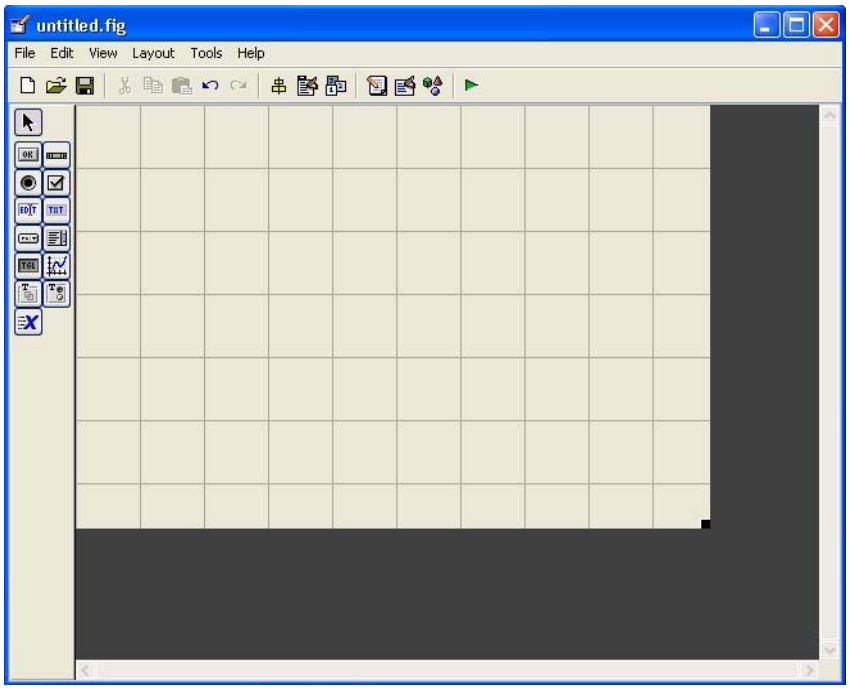


Figura 1 Janela de interface do GUIDE

1.4 Notas

O manual do programador da plataforma de simulação OSIP consiste numa abordagem profunda a todas as pastas, funções de execução gráficas, entre outros aspectos não menos importantes. Para além da descrição, e sempre que possível, as funções principais de construção do simulador são acompanhadas de um fluxograma explicativos da construção/programação do programa.

No final deste manual, em anexo, será apresentado um fluxograma geral da plataforma de simulação.

Capitulo 2

Sistema de directórios

A plataforma de simulação OSIP tem os seus ficheiros distribuídos por um conjunto de directórios no sentido de hierarquizar e separar os diferentes tipos de ficheiros do simulador.

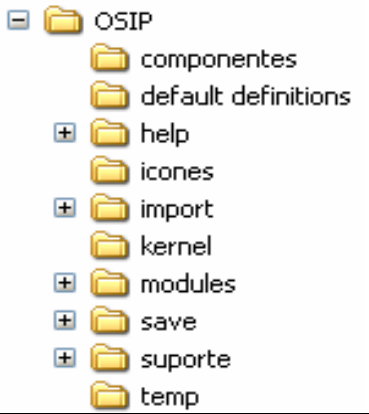


Figura 2.1 Estrutura de directórios do simulador

Atendendo à figura anterior podemos verificar que o simulador é constituído por um conjunto de 10 directórios distribuídos de seguinte forma:

2.1 Directório "Componentes"

O directório "*componentes*" contém ficheiros de texto com todas as informações necessárias sobre os portos dos componentes existentes no simulador. Por cada componente existente no simulador existe um ficheiro cujo nome é o do componente.

Dentro de cada ficheiro existem 9 campos. Para o exemplo consideremos o componente *attenuator*:

attenuator	Nome do componente
Passive§Components	Grupo de componentes a que pertence. Também corresponde ao nome do directório onde se encontra a *m-file* do componente.
1 optical_x	Número e tipo dos portos de entrada do componente.
1 optical_x	Número e tipo dos portos de saída do componente.
0	Número e tipo dos portos *Control Hi* do componente.
0	Número e tipo dos portos *Control Lo* do componente.
-1	Prioridade atribuída por defeito ao componente.
1	Campo que indica se é possível alterar o número de portos de entrada. '1'–> Pode-se alterar '0'–> Não se pode alterar
1	Campo que indica se é possível alterar o número de portos de entrada. '1'–> Pode-se alterar '0'–> Não se pode alterar

De notar que nos campos 3º,4º,5º e 6º têm o seguinte formato:

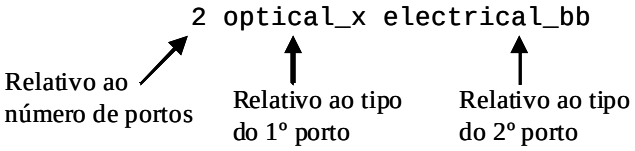


Figura 2.2 Formato utilizado nos ficheiros de definição dos portos dos componentes

No caso do número de portos ser nulo, então o campo inicial é zero e os restantes não aparecem.

2.2 Directório "Default Defenitions"

O directório "*Default Definitions*" contém ficheiros de texto com os valores por defeito dos parâmetros dos componentes existentes no simulador. Por cada componente existente no simulador existe um ficheiro cujo nome é o do componente.

Dentro de cada ficheiro pode existir um número de campos variável, dependendo do número de parâmetros do componente.

2.3 Directório "Help"

Neste directório estão os ficheiros de ajuda do simulador em formato *html*. Por cada ficheiro de ajuda existe um directório associado contendo ficheiros de imagens e formatações da página *html*.

2.4 Directório "Ícones"

O directório "*Ícones*" contém os ficheiros do tipo 'bmp' com as imagens correspondentes aos componentes do simulador. Cada ficheiro tem o nome do componente a que está associado. Existem ainda outros ficheiros do tipo 'bmp' que correspondem ao símbolo do simulador e à imagem adicionada por defeito caso o

utilizador adicione um componente ao simulador e não especifique a imagem associada ao componente.

2.5 *Directório* "Import"

Este directório contém as *m-files* associadas à importação dos componentes. Os ficheiros existentes são os necessários para efectuar o processo de importação de novos componentes ou exportar componentes existentes no simulador entre computadores.

2.6 *Directório "Kernel"*

O directório "*Kernel*" contém as *m-files* principais do simulador. Neste directório estão as *m-files* que criam a interface ("*osip_f.m*"), controlam o processo de *drag-and-drop* ("*mouseclickdown.m*", "*mouseclickup.m*", "*mousemovefnc.m*", "*position_mouse.m*", etc.) entre outras que permitem copiar, colar, apagar ícones na área de desenho, apagar as ligações dos portos, etc. É neste directório que se encontram as *m-files* que permitem criar os projectos e efectuar as simulações.

2.7 *Directório* "Modules"

Neste directório estão guardados os directórios de grupos de componentes do simulador:

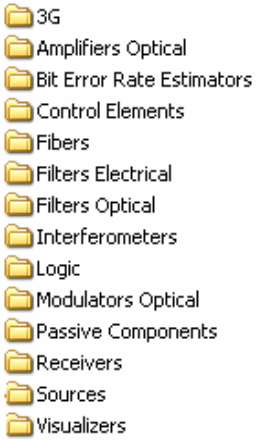


Figura 2.3 Directório Modules – onde se encontram os componentes do simulador

No directório "*Modules*" podemos encontrar 14 sub-directórios que correspondem ao grupo de componentes existentes no simulador. Estes directórios aparecem na interface de desenho do simulador. Dentro de cada directório estão guardadas as *m-files* dos componentes desse grupo.

2.8 Directório "Save"

O directório "*Save*" contém os ficheiros dos projectos guardados. São ficheiros com extensão 'osi'. Nesta fase estes ficheiros são de texto e contém o conteúdo de todas as variáveis e estruturas necessárias à retoma do projecto quando abertos para o simulador. Este directório é o directório de *save* por defeito, podendo os ficheiros serem guardados noutro local.

2.9 Directório "Suporte"

Neste directório pode-se encontrar os algoritmos e funções de suporte utilizados pelos componentes do simulador.

2.10 Directório "Temp"

Este directório encontra-se normalmente vazio. Serve de repositório de ficheiros temporários durante a criação do projecto e a execução da simulação. Nele são guardados ficheiros com as configurações dos portos e parâmetros por defeito, para o caso destes serem alterados durante a criação do projecto, e a *m-file* de execução da simulação.

Nos pontos seguintes são descritas de uma forma detalhada as funções mais importantes do simulador, por forma a se compreender como funciona todo o processo de construção do simulador, quais as estruturas de dados utilizadas e todas as funcionalidades de apoio à criação e execução dos projectos implementados no simulador.

Capitulo 3

Funções no directório Kernel

As funções no directório *Kernel*, tem como objectivo criar a interface gráfica do simulador (janela OSIP), activar e coordenar todos os eventos criados no simulador, por exemplo adicionar componentes, eliminar, copiar, colar, efectuar ligações entre os portos, etc.

3.1 Função OSIP

Esta é a função que inicia o arranque da plataforma de simulação OSIP.

Nesta função são declaradas todas as variáveis globais utilizadas no simulador e inicializadas as estruturas de dados.

3.1.1 Variáveis globais utilizadas no simulador:

flag_add_icon	*Flag* que é colocada a '1' enquanto decorre a operação de adicionar um novo ícone à área de desenho. Nas restantes situações encontra-se a '0'.
nicons	Estrutura que indica o número de ícones na área de desenho e o número de ícones de cada componente.
icons_schematic	Estrutura que contem informação dos ícones presentes na área de desenho.
flag_move	Flag que é colocada com o valor '1' enquanto decorre o evento de mover um ícone na área de desenho. Nas restantes situações encontra-se com o valor '0'.
ligacoes_icons	Estrutura que contém todos os parâmetros relacionados com as ligações dos ícones existentes na área de desenho.
flag_liga	Flag que é colocada com o valor '1' enquanto se realiza o evento de ligar dois portos. Nas restantes situações encontra-se com o valor '0'.
NUM	Estrutura que contém os parâmetros genéricos da simulação.
icons_valid	Array com a dimensão da estrutura *icons_schematic*. Este array indica quais as posições da estrutura *icons_schematic* que contêm os dados dos ícones existentes na área de desenho. Colocando o valor '1' se existir um ícone nessa posição e o valor '0' caso contrario (quando se apaga um ícone da área de desenho).
icon_liga	Estrutura que contém o índice do ícone na estrutura *icons_schematic*, o porto quando se pretende ligar e o tipo de porto (*in*, *out*, *control_H* e *control_L*).
timer1	*Timer* que periodicamente executa a função *position_mouse*

	para determinar qual a posição do cursor na janela do simulador e efectuar as devidas operações dependendo da posição onde este se encontra.
flag_select	Flag que é colocada com o valor '1' quando seleccionada uma área, na área de desenho.
BeginSelect	Array de duas posições que contém as coordenadas em *x* e em *y* do inicio da selecção na área de desenho.
PositionSelected	Array de 4 posições que contém as coordenadas em *x* e em *y* dos limites da selecção, nas primeiras 2 posições contém os limites em *x*, ordenados do limite inferior para o limite superior e nas 2 ultimas posições os limites em *y*, ordenados da mesma forma que em *x*.
IconsSelected	Array que contém os ícones seleccionados na área de desenho.
HandlesConectSelec	Estrutura que contém para cada ícone e para cada porto quais os *handles* das respectivas ligações que foram seleccionados na área de desenho.
icon_posi	Posição na estrutura *icons_schematic*, do ícone onde se encontra o cursor.
und	Estrutura de apoio a função UNDO
noise_flag	Variável global que indica o tipo da simulação em relação ao tratamento do ruído (numérica ou semi-analítica).
sem_lig	Estrutura de componentes que se encontram sem ligações com outros componentes quando se agrupa componentes.
data_out	Matriz de portos de entrada quando se agrupa componentes
data_out	Matriz de portos de saádaquando se agrupa componentes

3.1.2 Estruturas Utilizadas:

Estrutura icons_schematic:

A estrutura *icons_schematic* define completamente um ícone na área de desenho. Um ícone na área de desenho é constituído por uma imagem que identifica de certo modo o componente e pelos portos. Os portos podem ser: portos de entrada, saída

controlo inferior e controlo superior. Um ícone pode conter todos estes portos ou apenas alguns.

Cada ícone presente na área de desenho tem uma estrutura do tipo *icons_schematic* associada pelo que, ícones do mesmo componente podem ter parâmetros diferentes, estes parâmetros também se encontram guardados num ficheiro temporário que é criado no directório *temp* quando se adiciona um componente à área de desenho.

icons_schematic: Estrutura que contém informação dos ícones que se encontram na área de desenho.

nome	Nome do ícone quando é adicionado à área de desenho.
handle_imag	Array que contém os handles de um ícone na área de desenho.
icon_type	Indica de que tipo de componentes é o ícone.
posicao.x	Coordenada *x* do ícone na área de desenho (horizontal).
posicao.y	Coordenada *y* do ícone na área de desenho (vertical).
n_portos_in	Número de portos de entrada do ícone.
n_portos_out	Número de portos de saída do ícone.
n_control_H	Número de portos de controlo na parte superior do ícone.
n_control_L	Número de portos de controlo na parte inferior do ícone.
portos_in(i).tipo	Indica de que tipo é o porto 'i' de entrada.
portos_in(i).x	Indica a coordenada *x* do porto 'i' de entrada.
portos_in(i).y	Indica a coordenada *y* do porto 'i' de entrada.

portos_out(i).tipo	Indica de que tipo é o porto 'i' de saída.
portos_out(i).destination_port	Indica de que natureza é o porto de destino que se encontra ligado a este porto (*in*, *contol_H* ou *control_L*).
portos_out(i).x	Indica a coordenada *x* do porto 'i' de saída.
portos_out(i).y	Indica a coordenada *y* do porto 'i' de saída.
control_H(i).x	Indica a coordenada *x* do porto 'i' de controlo na parte superior do ícone.
control_H(i).y	Indica a coordenada *y* do porto 'i' de controlo na parte superior do ícone.
control_H(i).type	Indica o tipo do porto 'i' de controlo na parte superior do ícone.
control_L(i).x	Indica a coordenada *x* do porto 'i' de controlo na parte inferior do ícone.
control_L(i).y	Indica a coordenada *y* do porto 'i' de controlo na parte inferior do ícone.
control_L(i).type	Indica o tipo do porto 'i' de controlo na parte inferior do componente.
prioridade	Indica a prioridade do ícone na área de desenho.
change_port_in	Indica que se pode alterar o número de portos de entrada quando se encontra com o valor 1, caso se encontre com o valor 0 o número de portos de entrada do ícone é fixo.
change_port_out	Indica que se pode alterar o número de portos de saída quando se encontra com o valor 1, caso se encontre com o valor 0 o número de portos de saída do ícone é fixo.

Estrutura ligacoes_icons:

A estrutura *ligacoes_icons* contém todos os parâmetros das ligações entre os vários portos. Cada ligação de um porto tem associado um porto de destino, porto onde se termina a ligação, um percurso segundo as direcções horizontal (x) e vertical (y).

Quando um ícone é adicionado à área de desenho é criada uma posição na estrutura do tipo *ligacoes_icons* e inicializada para todos os portos (porto_in, porto_out, control_H, control_L) com os seguintes valores:

dest=" "

x=-1

y=-1

ligacoes_icons: Estrutura que contém todos os parâmetros das ligações dos ícones na área de desenho.

porto_in(i).dest	Contém a posição, na estrutura *icons_schematic*, do ícone que se encontra ligado ao porto de entrada '*i*'.
porto_in(i).x	É um array que contém na primeira e ultima posição, a coordenada *x* do porto de entrada e a coordenada *x* do porto a que se encontra ligado respectivamente. Nas restantes posições do array contém os handles das ligações.
porto_in(i).y	É um array que contém na primeira e ultima posição, a coordenada *y* do porto de entrada e a coordenada *y* do porto a que se encontra ligado respectivamente. Nas restantes posições do array contém os handles das ligações.
porto_out(i).dest	Contém a posição, na estrutura *icons_schematic*, do ícone que se encontra ligado ao porto de saída 'i'.
porto_out(i).x	É um array que contém na primeira e ultima posição, a coordenada *x* do porto de saída e a coordenada *x* do porto a que se encontra ligado respectivamente. Nas restantes posições do array contém os handles das ligações.
porto_out(i).y	É um array que contém na primeira e ultima posição, a coordenada *y* do porto de saída e a coordenada *y* do porto a que se encontra ligado respectivamente.

Nas restantes posições do array contém os handles das ligações.

control_H(i).dest Contém a posição, na estrutura *icons_schematic*, do ícone que se encontra ligado ao porto 'i' de controlo superior.

control_H(i).x É um array que contém na primeira e ultima posição, a coordenada *x* do porto de controlo superior e a coordenada *x* do porto a que se encontra ligado respectivamente.

Nas restantes posições do array contém os handles das ligações.

control_H(i).y É um array que contém na primeira e ultima posição, a coordenada *y* do porto de controlo superior e a coordenada *y* do porto a que se encontra ligado respectivamente.

Nas restantes posições do array contém os handles das ligações.

control_L(i).dest Contém a posição, na estrutura *icons_schematic*, do ícone que se encontra ligado ao porto 'i' de controlo inferior.

control_L(i).x É um array que contém na primeira e ultima posição, a coordenada *x* do porto de controlo inferior e a coordenada *x* do porto a que se encontra ligado respectivamente.

Nas restantes posições do array contém os handles das ligações.

control_L(i).y É um array que contém na primeira e ultima posição, a coordenada *y* do porto de controlo inferior e a coordenada *y* do porto a que se encontra ligado respectivamente.

Nas restantes posições do array contém os handles das ligações.

Estrutura NUM:

A estrutura *NUM* contém os parâmetros genéricos da simulação estes parâmetros são utilizados por vários componentes quando se executa uma simulação, mas também para configurar certos parâmetros dos componentes.

NUM Valor por defeito dos parâmetros numéricos da simulação

BR	Bit Rate	[Gbit/s]
fa	Frequência de amostragem	[Hz]
fref	Frequência de referencia do sistema	[THz]
NBITS	Nº de bits da sequencia de informação	
npts	Nº de pontos da sequencia de informação	
f	Vector de frequências cuja frequência central é a frequência de referencia do sistema.	[Hz]
t	Vector de tempo correspondente ao vector anterior	[s]

Valor por defeito dos parâmetros numéricos da simulação:

NUM Valor por defeito dos parâmetros numéricos da simulação

NUM. BR	10	[Gbit/seg]
NUM. fa	160e9	[Hz]
NUM. fref	193.55	[THz]
NUM.NBITS	512	
NUM. npts	8192	

Os valores dos campos NUM.t e NUM.f são calculados de acordo com as seguintes equações:

```
NUM.f = NUM.fref +(0:UM.npts-1)*NUM.fa/NUM.npts - NUM.fa/2;
```

NUM.t = (0:NUM.npts-1)*(1/NUM.fa);

Estrutura nicons:

A estrutura *nicons* é uma estrutura dinâmica que contém o número de ícones totais na área de desenho, mas também o número de ícones de cada tipo.

Quando se selecciona um componente pela primeira vez, este é adicionado à estrutura.

nicons: Estrutura que contém o número de ícones totais e o número de ícones de cada tipo de componentes presentes na área de desenho.

total Número total de ícones presentes na área de desenho.

'Nome do componente' Os ícones de cada tipo de componente são adicionados automaticamente a estrutura quando são seleccionados pela primeira. Esta estrutura cresce automaticamente. O número total de ícones na área de desenho e o número de ícones de cada tipo de componentes são incrementados quando se adiciona um novo componente à área de desenho.

Estrutura icon_liga:

Esta estrutura é actualizada quando se coloca o cursor sobre um porto de um ícone na área de desenho. Tem como finalidade indicar qual o porto do ícone que se pretende ligar.

icon_liga: Estrutura que indica o porto do ícone que se pretende ligar assim como o tipo de porto.

icon_liga.posi Índice do ícone, na estrutura *icons_schematic*, que se pretende ligar.

icon_liga.porto Índice do porto onde se pretende efectuar a ligação.

icon_liga.type Tipo do porto ('*in*', '*out*', '*control_H*' e '*control_L*').

Ver também: *osip_f, mouseclickdown, mouseclickup, keypressfunc e resizefcn.*

Help Matlab:

– 'Timer'

– 'handles'

3.1.3 Descrição do algoritmo:

Inicialmente são declaradas e inicializadas todas as variáveis globais utilizadas no simulador para o controlo e gestão das diversas operações que se podem realizar no simulador.

De seguida são adicionados os vários directórios, utilizados pelo simulador, ao conjunto de directórios do Matlab através da função *addpath* do Matlab. Deste modo é possível executar funções que se encontram em diferentes directórios e em simultâneo. Como a pasta "*help*" possui uma grande quantidade de sub-directórios achou-se conveniente retirá-la deste processo por não ser necessária ao funcionamento do OSIP enquanto simulador.

A seguir é executada a função *osip_f*. Esta função cria a janela de interface gráfica do simulador OSIP, janela onde podem ser criados os projectos a simular. Todas as funções associadas aos menus e aos botões acedíveis a partir da janela de interface (*open*, *save*, *undo*, *zoon in*, *zoon out,* etc.) estão referenciadas ou implementadas na função *osip_f.*

Por fim activa todas as funções que controlam e coordenam todos os eventos criados no simulador como por exemplo adicionar componentes, efectuar ligações entre os diversos portos, apagar ícones, etc.

3.1.4 Fluxograma da m-file OSIP

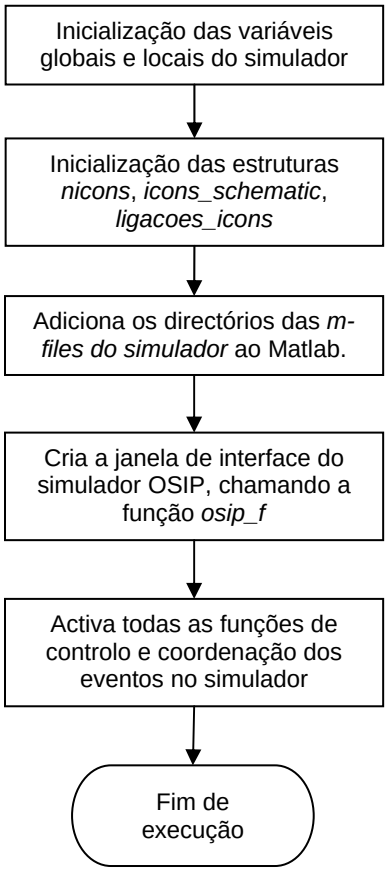


Figura 3.1 Fluxograma da m-file OSIP

3.2 Função *position_mouse*

Esta função determina periodicamente, intervalos de tempo indicados pelo timer1, qual a posição do cursor no simulador. Dependendo da posição do cursor no simulador, é alterado o formato do cursor da seguinte forma:

- Quando se pretende fazer zooming da área de desenho o cursor toma a forma de um quadrado
- Quando o cursor se encontra sobre um componente na área de desenho este tem a forma "*fleur*";
- Quando o cursor se encontra sobre um porto de um componente na área de desenho este tem a forma "*circle*";

- Quando o cursor se encontra apenas sobre a área de desenho, este tem a forma "*cross*";
- Quando o cursor se encontra fora da área de desenho, mas sobre a janela para adicionar um novo componente ao schematic, o cursor tem a forma "*fleur*";
- Em todos os outros casos o cursor tem a forma normal "*arrow*";

A forma do cursor mantém-se sempre a mesma – "*crosshair*" – enquanto se estiver a estabelecer uma ligação entre dois portos.

3.2.1 Utilização:

- position_mouse(handles)

Parâmetros de entrada

handles: Estrutura que contém todos os handles dos objectos que formam a janela de interface do simulador.

Parâmetros de saída:

Não existem.

Variáveis Globais:

flag_add_icon

icons_schematic

nicons

flag_move

icon_liga

flag_liga

ligacoes_icons

icons_valid

flag_ZoomOut

flag_ZoomIn

icon_posi

Esta função altera o aspecto do cursor tendo em conta as *flags* globais que indicam o estado de determinadas operações.

Ver também: *OSIP, osip_f.*
Help Matlab:

– Figure Pointer
– handles

3.2.2 Descrição do algoritmo:

Inicialmente é determinada as coordenadas do cursor na janela do simulador, a posição janela do simulador, o tamanho da área de desenho e os limites da área de desenho x e y. Verifica-se então se o cursor se encontra dentro da área de desenho. Se sim, verifica se as *flags* que indicam que se está a fazer o *zooming* da área de desenho (*flag_ZoomOut* e *flag_ZoomIn*) estão activas. Se estiverem activas, o cursor fica com o aspecto de um quadrado.

Caso as *flags* de *zooming* estejam inactivas então o cursor toma a forma de cruz ("*cross*") e na barra de estado aparece a seguinte mensagem: "*OSIP schematic*". Se o cursor se encontra sobre um ícone da área de desenho então o cursor toma a forma de dupla seta em cruz ("*fleur*") e a barra de estados apresenta a mensagem "*Name icon -> ,<nome do ícone no schematic>, - double clik to edit property box*".

No caso do cursor se encontrar sobre um porto (entrada, saída ou controlo) o cursor toma a forma de círculo ("*circle*"), aparecendo na barra de estado a mensagem: *"'Signal <tipo – imput, output ou control> ',' - Port', <numero do porto>' - type ','->', <tipo de sinal>'".*

Ao se iniciar uma ligação o cursor toma a forma de cruz fina ("*crosshair*"), permanecendo este cursor enquanto a *flag_liga* estiver a 1, ou seja, enquanto não se terminar a ligação ou esta não for interrompida pela tecla *Escape*.

Caso o cursor não se encontra na área de desenho, mas dentro da área de pré-visualização da imagem do ícone do componente, então o cursor toma a forma de dupla seta em cruz ("*fleur*"). Na barra de estado aparece a mensagem: "*Add new icon*".

Se o cursor não se encontrar na área de desenho ou na área de pré-visualização, então a forma do cursor é a típica seta ("arrow") e na barra de estado aparece a mensagem "*Select component*".

3.2.3 Fluxograma da função position_mouse:

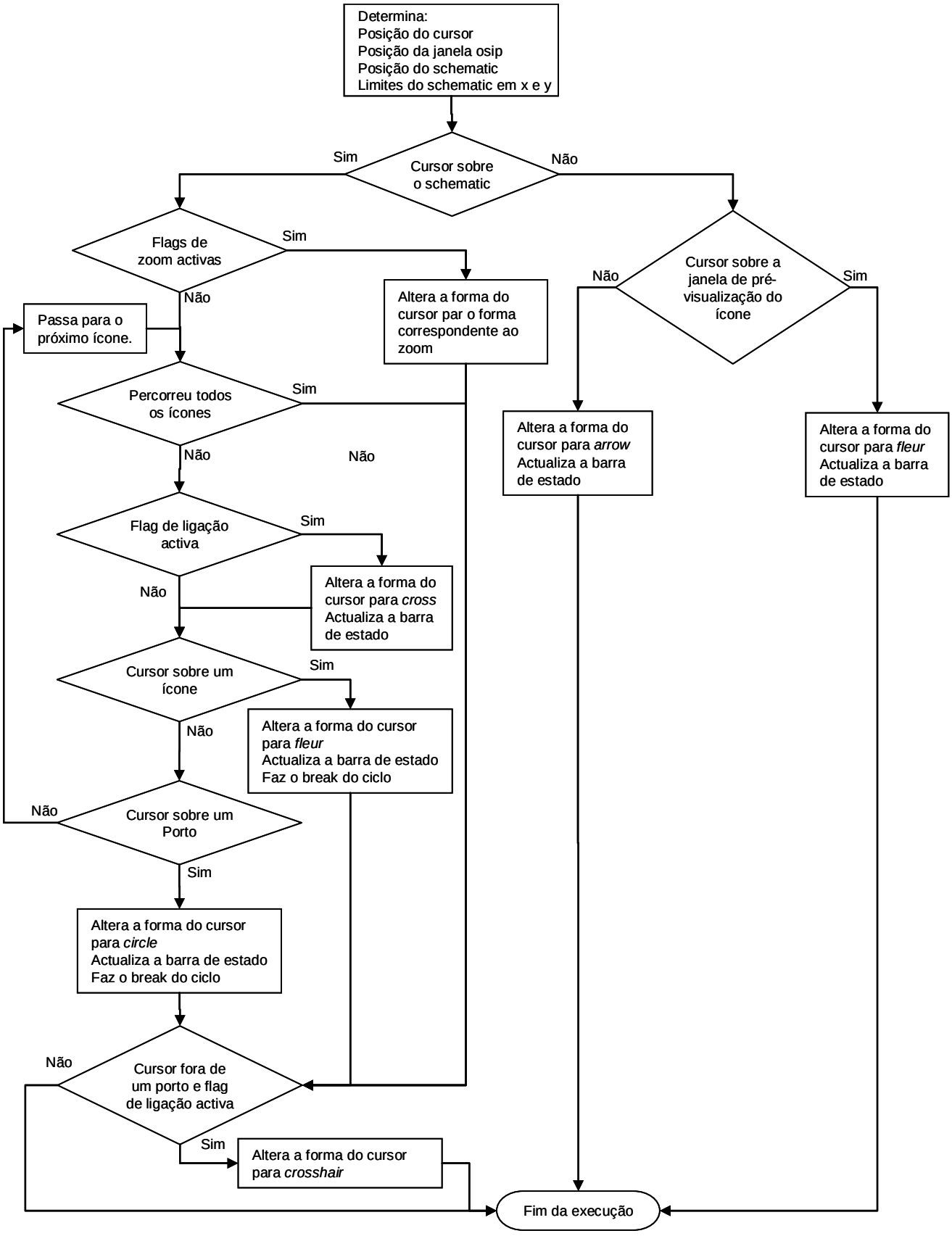


Figura 3.2. Fluxograma da função position_mouse

3.3 Função *mouseclickdown*

Esta função é executada quando se pressiona o botão do rato na janela do simulador. Dependendo do botão pressionado (direito ou esquerdo) e da posição na janela do simulador onde este foi pressionado, são executados diferentes eventos.

3.3.1 Utilização:

- mouseclickdown(handles)

Parâmetros de entrada:

handles: Estrutura que contém os *handles* dos objectos da janela do simulador OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

flag_add_icon

nicons

icons_schematic

flag_move

icon_posi

flag_liga

icon_liga

Ligacoes_icons

icons_valid

flag_select

BeginSelect

PositionSelected

IconsSelected

flag_ZoomOut

flag_ZoomIn

und

Ver também: *mouseclickup*, *mousemovefnc*, *conect_port*, *IconMoveFnc*, *select*.
Help Matlab:

– 'SelectionType',

– 'Button'.

3.3.2 Descrição do algoritmo:

Inicialmente começa-se por determinar qual a posição e a dimensão da janela de pré visualização dos ícones e da área de desenho. De seguida determina-se as coordenadas do cursor na janela do simulador (janela *OSIP*) assim como os limites da área de desenho em *x* e em *y*.

De seguida determina-se qual o botão do rato que foi pressionado, através do seguinte comando: get(handles.osip_f,'SelectionType'), este comando devolve uma string que pode tomar três valores diferentes:

- 'normal' indica que foi pressionado o botão esquerdo do rato.
- 'alt' indica que foi pressionado o botão do lado direito do rato.
- 'open' indica que foi efectuado um duplo click.

Quando o botão esquerdo do rato é pressionado sobre a janela de pré visualização dos ícones, elimina-se qualquer selecção que exista no esquemático, activa-se a flag que indica que se pretende adicionar um componente à área de desenho (*flag_add_icon=1*), determina-se qual o componente que se encontra na janela de pré visualização, coloca-se a sua imagem num *axes* que se encontra atrás desta janela, este *axes* não é visível pelo utilizador enquanto não se mover o componente para a área de

desenho; e activa-se a função *mousemovefnc* através do comando: set(handles.osip_f,'WindowButtonMotionFcn','mousemovefnc(handles)'). Esta função desloca o *axes*, que contém a imagem do componente que se pretende adicionar à área de desenho, quando se pressiona o botão do rato na janela de pré visualização e se desloca o cursor com o botão do rato pressionado para a área de desenho.

No caso do botão do rato ser pressionado na área de desenho são desencadeados vários eventos dependendo das seguintes situações:

Se alguma das *flags* de zoom se encontra activa (flag_ZoomIn==1 ou flag_ZoomOut==1) e para o caso de se ter efectuado um duplo click, é eliminado o zoom da área de desenho. Se for efectuado apenas um click com uma destas *falgs* activa é efectuado o zoom *in* ou zoom *out* na área de desenho dependendo da *flag* que se encontra activa. Para efectuar o zoom *in* chama-se a função FuncZoomIn e para efectuar o zoom *out* a função FuncZoomOut.

Se as *flags* de zoom se encontrarem desactivadas, verifica-se qual é o tipo de cursor que se encontra no esquemático através do seguinte comando: get(handles.osip_f,'Pointer'), este comando devolve uma string com o tipo de cursor.

Se o cursor for do tipo '*circle*', indica que este se encontra sobre um porto de um ícone. Como tal verifica-se se o porto onde se efectuou o click já se encontra ligado, em caso afirmativo envia-se uma mensagem de erro ao utilizador a indicar que o porto já se encontra ligado. Caso contrário é activada a *flag* que indica que se vai efectuar uma ligação (flag_liga=1) e activa-se a função que permite ligar dois portos, através do comando: set(handles.osip_f,'WindowButtonMotionFcn','conect_port(handles)').

Se o cursor for do tipo '*fleur*', sabe-se que o cursor se encontra sobre um ícone na área de desenho. No caso de ter sido efectuado um duplo click sobre o ícone, string com o do botão do rato pressionado com o valor '*open*', desactiva-se a flag de mover um ícone flag_move=0, e abre-se a janela de configuração dos parâmetros do componente através do seguinte comando: feval(file_name, {icons_schematic(i).nome}, i, handles.osip_f,

handles.schematic). Onde file_name, é o nome da janela de configuração dos parâmetros do componente, icons_schematic(i).nome, nome do ícone no esquemático, i, índice do ícone na estrutura *icons_schematic*, handles.osip_f handles da janela do simulador e handles.schematic handle da janela da área de desenho. Caso se efectue apenas um click sobre o ícone, é activada a flag de mover um ícone flag_move=1, verifica-se se apenas se pretende mover um ícone ou um conjunto de ícone previamente seleccionados e activa-se a função de mover os ícones através do seguinte comando: set(handles.osip_f,'WindowButtonMotionFcn', 'IconMoveFnc(handles, IconsSelected,icon_posi)'), onde handles é uma estrutura que contém os handles da janela do simulador, IconsSelected, array que contém os índices na estrutura *icons_schematic* dos ícones seleccionados e icon_posi é a posição do ícone, na estrutura icons_schematic, que se efectuou o click. Nesta situação ainda é chamada a função *undo_function.m* que irá realizar a salvaguarda do projecto para um possível retrocesso. (ver função *undo* – capítulo 3.23)

Se o cursor é do tipo '*cross*', indica que este se encontra na área de desenho, mas fora de qualquer porto e ícone. Como tal elimina-se qualquer selecção que exista, desactiva-se a função de copy, a função de agrupamento de componentes, do menu *Edit* através do comando: set(handles.copy,'Enable','off'), a função de agrupamento de componentes (*Creat component*), do menu *tools* com o comando: set(handles.creat_component_hierarchy,'Enable','off'). No caso de se efectuar apenas um click, string com o botão do rato pressionado do tipo '*normal*', determina-se as coordenadas do click na área de desenho e activa-se a função de selecção no com o seguinte comando: set(handles.osip_f,'WindowButtonMotionFcn','select(handles)'). Caso se efectue um duplo click abre-se a janela de configuração dos parâmetros genéricos da simulação generic_parameters.

Por fim se for pressionado o botão do lado direito do rato no esquemático, elimina-se qualquer selecção existente no esquemático e desactiva-se a função de copy no menu *Edit* do simulador através do comando: set(handles.copy,'Enable','off').

3.3.3 Fluxograma da função mouseclickdown

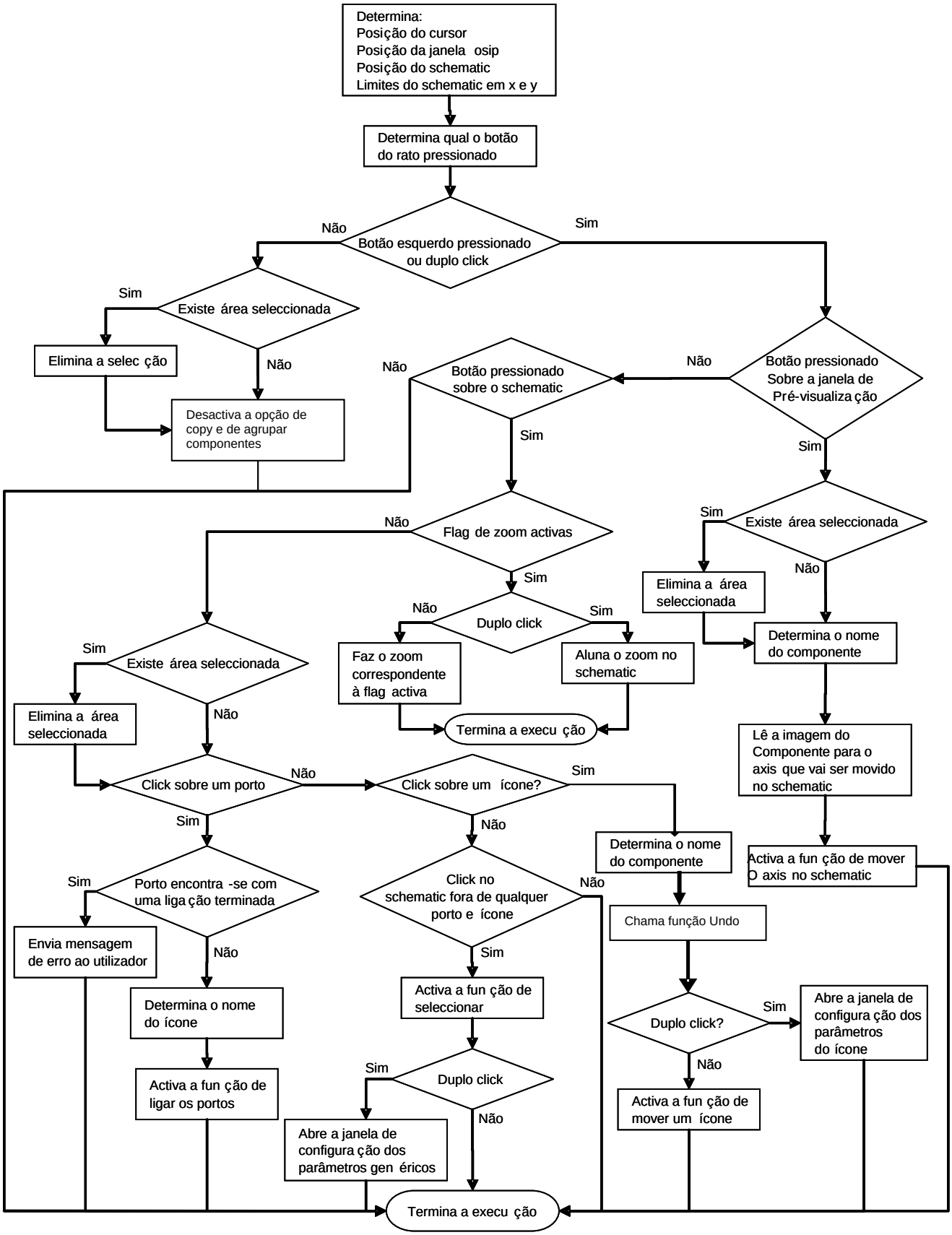


Figura 3.3. Fluxograma da função mouseclickdown

3.4 Função *mouseclickup*

Esta função é executada quando se faz o *up* do botão do rato no simulador. As operações que são desencadeadas dependem do estado das *flags* e da posição do cursor onde é efectuado o *up*.

3.4.1 Utilização:

- mouseclickup(handles)

Parâmetros de entrada:

handles: Estrutura que contém os handles dos objectos da janela do simulador OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

flag_add_icon

nicons

icons_schematic

flag_move

icon_posi

flag_liga

icon_liga

ligacoes_icons

icons_valid

flag_select

PositionSelected

IconsSelected

HandlesConectSelec

Ver também: *LigarIcon, mouseclickdown, osip, HandlesConectSelec*

Help Matlab:

– *WindowButtonMotionFcn*,

– *UIContextMenu*.

3.4.2 Descrição do algoritmo:

Inicialmente determina-se as coordenadas do cursor na área de desenho, a posição da janela do simulador no ecrã, a posição da área de desenho na janela do simulador e os limites da área de desenho em *x* e em *y*.

De seguida desactivam-se as funções relacionadas com mover o cursor na janela do simulador com o botão do rato pressionado através do comando: set(handles.osip_f,'WindowButtonMotionFcn','').

Depois verifica-se se existem ícones seleccionados, através da variável IconsSelected, em caso afirmativo determina-se os valores em *x* e *y* (*xmax*, *xmin*, *ymax* e *ymin*) dos ícones que delimitam a área de selecção. Se estes valores excederem os limites da área de desenho é chamada a função IconMoveFnc(handles,IconsSelected,icon_posi ,deltax,deltay) para efectuar um deslocamento deltax segundo *x* e deltay segundo *y* dos ícones seleccionados para os colocar dentro da área de desenho.

Se a flag de selecção se encontrar activa flag_select==1, indica que se seleccionou uma área na área de desenho, neste caso vai se determinar quais os ícones seleccionados, guardando-se os índices dos respectivos ícones na variável global IconsSelected. De seguida determina-se quais os handles das ligações dos ícones seleccionados que se encontram dentro da área seleccionada, cujos ícones dos portos de

destino se encontram fora da área de selecção. Para tal é chamada a função HandlesConections, que guarda os respectivos handles na variável global HandlesConectSelec. De realçar que, quando na área de desenho se encontraram mais do que um componente é activada no simulador a possibilidade de se agrupar componentes, através do comando: set(handles.creat_component_hierarchy,'Enable','on');

No caso da flag que indica que se pretende adicionar um componente à área de desenho se encontra activa flag_add_icon==1, lê-se o nome do novo componente que se encontra no directório ..*temp*, no ficheiro *novo_icon.txt*. De seguida coloca-se a imagem deste no local dado pelo cursor na área de desenho, actualiza-se a estrutura nicons, onde é incrementado o número de ícones totais na área de desenho, assim como o número de ícones deste componente, activa-se o *context menu*, são as operações disponíveis quando faz um click com o botão direito do rato no ícone. Depois são lidos todos os parâmetros por defeito do componente, este parâmetros encontra-se no directório ..*default definitions*, e as suas características quanto ao número e tipo de portos que se encontram no directório ..\componentes. São desenhados os portos do componente e guardando os respectivos parâmetros na estrutura *icons_schematic*.

A posição dos portos é determinada de forma automática através das seguintes expressões:

Para os portos de entrada.

$$P_x = P_{x\_cursor} - 25 - 14$$

$$P_y = P_{y\_cursor} - 25 + i \cdot \frac{50}{n\_ports\_in + 1}$$ Onde *i* indica o índice do porto de entrada.

Para os portos de saída

$$P_x = P_{x\_cursor} + 25 + 14$$

$$P_y = P_{y\_cursor} - 25 + i \cdot \frac{50}{n\_ports\_out + 1}$$ Onde *i* indica o índice do porto de saída.

Para os portos de controlo superior.

$$P_x = P_{x\_cursor} - 25 + i \cdot \frac{50}{n\_ports\_control\_H + 1}$$ Onde *i* indica o índice do porto de controlo superior.

$$P_y = P_{y\_cursor} + 25 + 14$$

Para os portos de controlo inferior.

$$P_x = P_{x\_cursor} - 25 + i \cdot \frac{50}{n\_ports\_control\_L + 1}$$ Onde *i* indica o índice do porto de controlo inferior.

$$P_y = P_{y\_cursor} - 25 - 14$$

Por fim é actualizada a estrutura *ligacoes_icons*, colocando uma nova posição para o ícone adicionado.

Se a flag que indica que se está a efectuar uma ligação está activa flag_liga==1, é chamada a função para terminar uma ligação *LigarIcon(icon_liga,handles)* onde é dado como entrada a estrutura *icon_liga* que contém o índice do ícone, na estrutura *icons_schematic*, de início da ligação, o índice do porto de início da ligação e o tipo do porto.

3.4.3 Fluxograma da função mouseclickup:

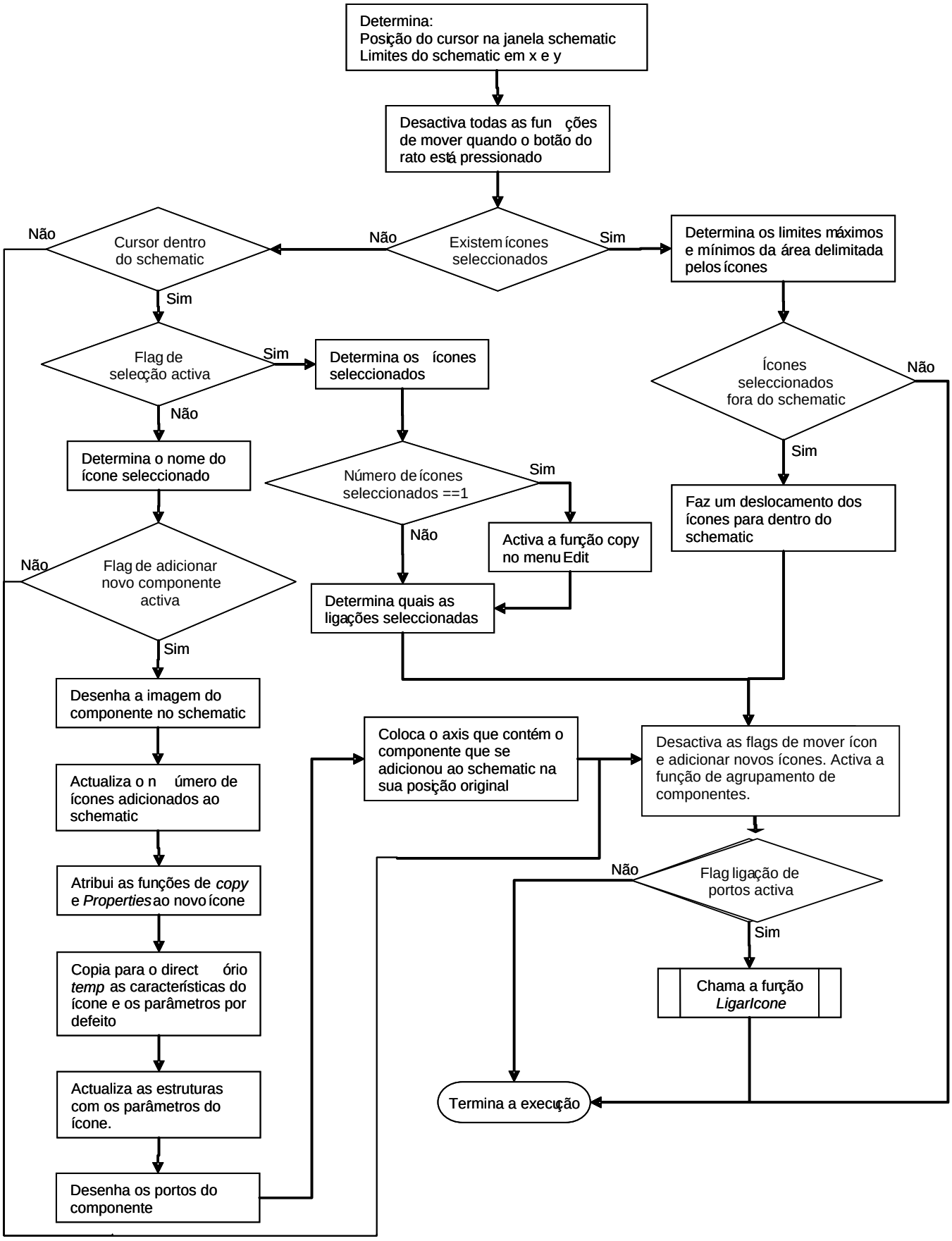


Figura 3.4 Fluxograma da função mouseclickup

3.5 Função *mousemovefnc*

Esta função faz o deslocamento do *axes*, que se encontra por detrás da janela de pré-visualização e contém a imagem do componente que se pretende adicionar à área de desenho, para a posição dada pelo cursor dentro da área de desenho quando se move o rato com o botão esquerdo pressionado, depois de se ter efectuado um click na janela de pré-visualização.

Quando se faz o *up* do botão do rato dentro da área de desenho é chamada a função mouseclickup(handles) para desenhar a imagem do componente e os respectivos portos na local dado pelo cursor.

3.5.1 Utilização:

- mousemovefnc(handles)

Parâmetros de entrada:

handles: Estrutura que contém os *handles* dos objectos da janela do simulador OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

Não existem.

3.5.2 Descrição do algoritmo:

Inicialmente determina-se as coordenadas do cursor na área de desenho, a posição da área de desenho dentro da janela OSIP e os limites em x e y.

De seguida verifica-se se o cursor se encontra dentro da área de selecção, em caso afirmativo coloca-se o *axes*, que contém a imagem do componente que se pretende adicionar à área de desenho, no local onde se encontra o cursor.

3.6 Função *HandlesConenctions*

Esta função é executada quando se faz o up do botão do rato depois de se seleccionar uma área no esquemático.

A finalidade desta função é determinar qual os troços das ligações que se encontram dentro da área de selecção, para ligações que se iniciam dentro da área de selecção e termina fora da área de selecção ou vice-versa.

3.6.1 Utilização:

- HandlesConectSelec=HandlesConections(icon_index, icons_schematic,ligacoes_icons,IconsSelected, PositionSelected,HandlesConectSelec)

Parâmetros de entrada:

icon_index	Índice do ícone para o qual se pretende determinar as ligações que terminam fora da área de selecção.
icons_schematic:	Estrutura que contem informação dos ícones presentes na área de desenho.
ligacoes_icons:	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.
IconsSelected	Array com os ícones seleccionados.
PositionSelected	Array que indica qual a área seleccionada.
HandlesConectSelec	Estrutura que contém os handles dos troços das ligações que se encontram dentro da área de selecção, para ligações que se iniciam dentro da área de selecção e terminam num porto de um ícone fora da área de selecção.

Parâmetros de saída:

HandlesConectSelec "Estrutura explicada nos parâmetros de entrada".

Variáveis Globais:

Não existem.

Estrutura HandlesConectSelec:

HandlesConectSelec(i).in(j).handle	Handles dos troços das ligações dos portos de entrada '*in*'.
HandlesConectSelec(i).out(j).handle	Handles dos troços das ligações dos portos de saída '*out*'.
HandlesConectSelec(i).control_H(j).handle	Handles dos troços das ligações dos portos de controlo superior '*control_H*'.
HandlesConectSelec(i).control_L(j).handle	Handles dos troços das ligações dos portos de controlo inferior '*control_L*'.

Nota: 'i' indica a posição do ícone na estrutura *icons_schematic* e 'j' o índice do porto da ligação.

Ver também: *mouseclickup, mouseclickdwon, IconMoveFnc*

Help Matlab:

– 'Get'

– 'Handles'

3.6.2 Descrição do algoritmo:

Algoritmo testa todos os portos do ícone começando pelos portos de entrada. Para todos os portos de entrada verifica: se o porto tem destino (o destino é a posição na estrutura *icons_schematic* do ícone onde a ligação termina), se o porto tiver destino determina qual é o índice do porto de destino da ligação.

Para o caso da ligação estar terminada, ou seja, se existir um destino da ligação, verifica-se se o ícone de destino se encontra dentro da área de selecção. Caso o ícone de

destino se encontre dentro da área de selecção, coloca-se o valor '0' na estrutura *HandlesConectSelec* correspondente ao ícone da respectiva ligação do porto de entrada HandlesConectSelec(icon_index).in(j).handle=0.

Se o ícone de destino se encontrar fora da área de selecção quer dizer que se está na situação em que a ligação tem origem num porto dentro da área de selecção e termina num porto fora da área de selecção. Neste caso determina-se quais os troços da ligação que se encontram dentro da área de selecção e guarda-se os respectivos handles na estrutura *HandlesConectSelec* tendo em conta qual o índice do ícone e o respectivo porto HandlesConectSelec(icon_index).in(j).handle(end)= ligacoes_icons (icon_index).porto_in(j).x(k) onde 'icon_index' é o índice do ícone, 'j' o índice do porto e 'k' identifica o troço da ligação na estrutura *ligacoes_icons*.

Este procedimento é aplicado para todos os portos ('*in*', '*out*', '*control_H*', '*control_L*').

3.6.3 Fluxograma da função HandlesConections:

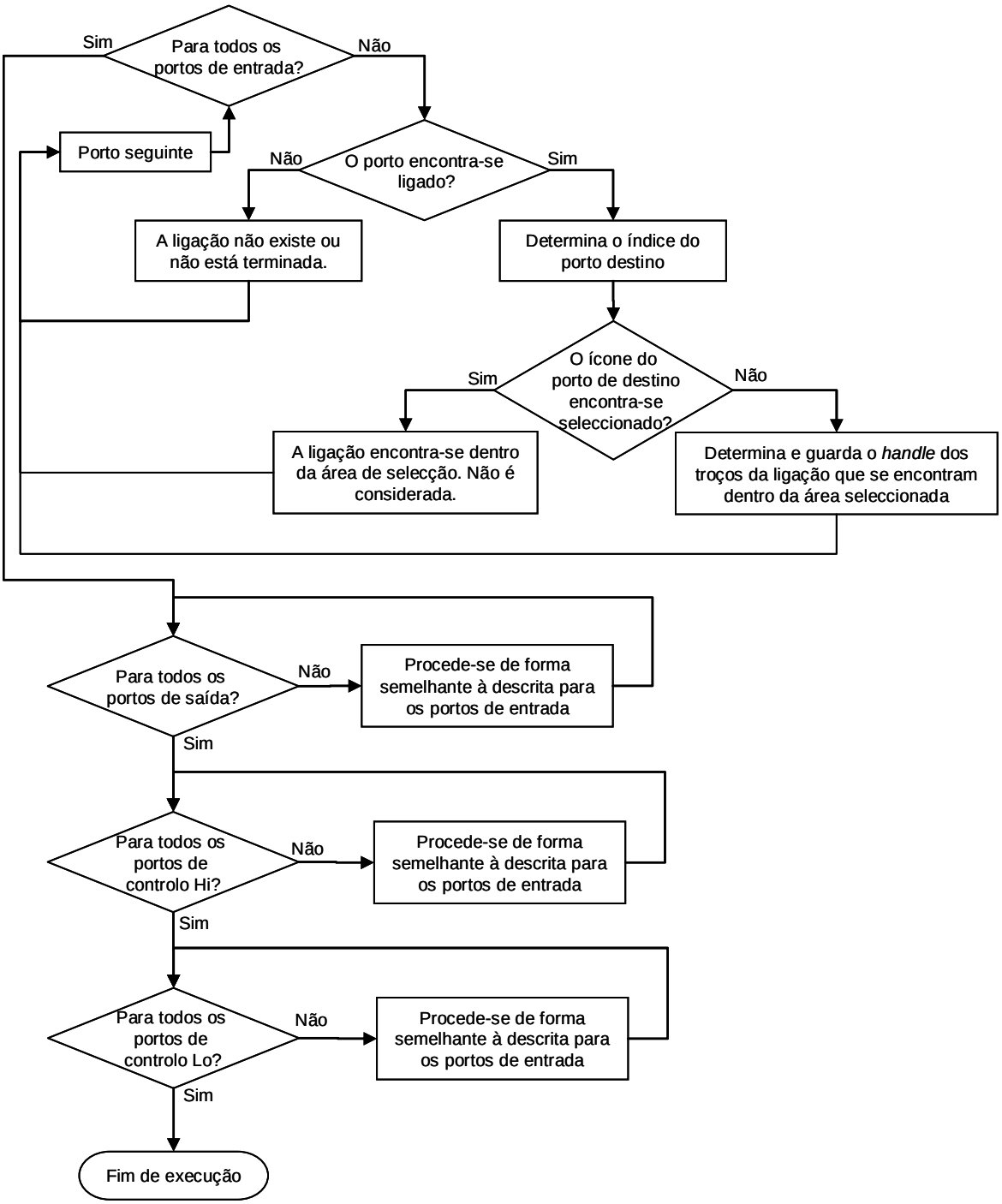


Figura 3.6 Fluxograma da função HandlesConections

3.7 Função *LigarIcon*

Esta função é executada para terminar uma ligação entre dois portos, ou seja quando é efectuado o *up* do botão esquerdo do rato quando este se encontra sobre um porto e a *flag* que indica que se está a efectuar uma ligação se encontra activa flag_liga==1.

3.7.1 Utilização:

- LigarIcon(icon_liga,handles)

Parâmetros de entrada:

icon_liga	Estrutura que contém o índice do ícone, o índice do porto e o tipo do ícone onde se iniciou a ligação.
icon_liga.posi	Posição do ícone na estrutura icons_schematic.
icon_liga.porto	Índice do porto seleccionado.
icon_liga.type	Tipo do porto seleccionado.
'in'	Porto de entrada.
'out'	Porto de saída.
control_H	Porto de controlo superior.
control_L	Porto de controlo inferior.

Parâmetros de saída:

Não existem.

Variáveis Globais

icons_schematic

ligacoes_icons

icons_valid

flag_liga

und

3.7.2 Descrição do algoritmo:

Inicialmente começa-se por salvaguardar o esquemático, através da evocação da função *undo_function* com a flag *und.clic* activa, e determina-se as coordenadas do cursor

na janela do simulador, a posição da janela no ecrã, a posição da área de desenho na janela do simulador e os limites em *x* e *y* da área de desenho.

De seguida determina-se qual o tipo do porto de inicio da ligação através do parâmetro *icon_liga.type*.

Caso o tipo do porto de início da ligação seja um porto de entrada *icon_liga.type* ==‘in’, verifica-se para todos os ícones presentes na área de desenho se o *up* do botão do rato foi sobre um porto de saída de um ícone. Quando se verificar esta condição, é feito um teste para verificar se o porto onde se pretende efectuar a ligação é do mesmo tipo que o porto de início da ligação. Se os portos forem de diferentes tipos é enviada uma mensagem de erro através do seguinte comando errordlg('this connection is not possible','!! ERROR!!') e apagada o troço da ultima ligação.

Caso os portos sejam do mesmo tipo verifica-se se o porto de saída onde se pretende efectuar a ligação já se encontra ligado a um outro porto, em caso afirmativo é enviado uma mensagem de erro a informar que o porto já se encontra ligado. Caso contrário é apagada, caso exista a ligação a este porto que não tenha sido terminada; é actualizada a estrutura *icons_schematic* com o valor icons_schematic(i).portos_out(j).destination_port='in' onde *j* indica o índice do porto de saída, pois os portos de saída podem ter ligações provenientes de portos de entrada ao portos de controlo e como tal temos de fazer a distinção (o parâmetro destination_port só existe para os portos de saída); é guardado no parâmetro *dest* da estrutura *ligacoes_icons* a posição do ícone na estrutura *icons_schematic* tanto para o porto de entrada como para o porto de saída, é ajustada a ligação ao porto de saída apagando a ligação existente e efectuado uma nova ligação tendo em conta a posição do porto no esquemático.

Por fim é desactivada a função de ligar os portos através do seguinte comando: set(handles.osip_f,'WindowButtonMotionFcn','');

A explicação anterior refere-se ao exemplo de uma ligação entre um porto de entrada e um porto de saída, mas o raciocínio aplica-se as restantes combinações de ligações, com a diferença que para os portos de entrada e de controlo não existe o campo *destination_port* na estrutura *icons_schematic*, pois este portos só podem ser ligados a portos de saída.

3.7.3 Fluxograma da função LigarIcon:

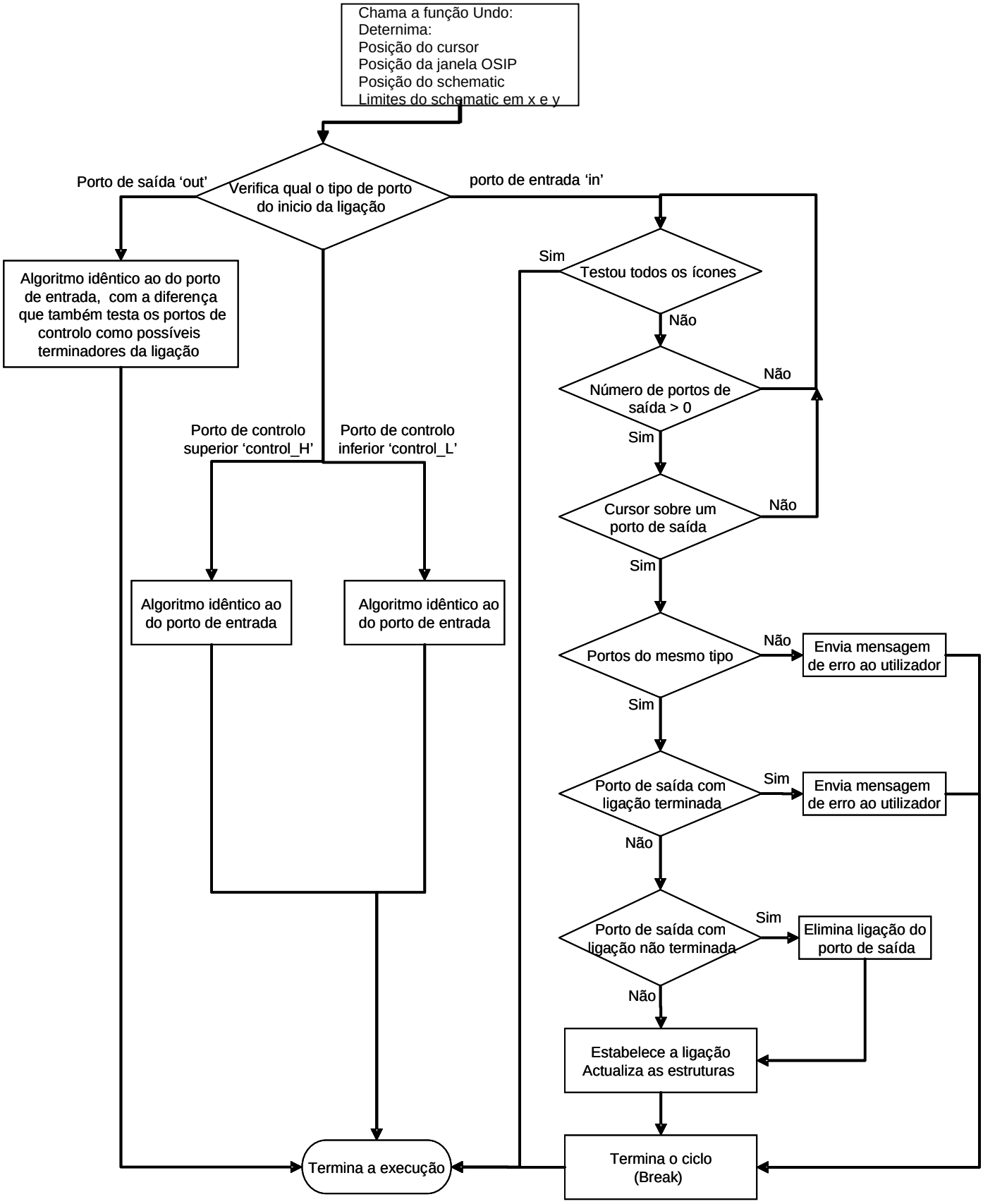


Figura 3.7 Fluxograma da função LigarIcon

3.8 Função *IconMoveFnc*

Esta função é executada quando se pretende move um ícone ou conjunto de ícones e ligações na área de desenho.

Esta função é invocada automaticamente quando se desloca o cursor na área de desenho depois de se clicar num ícone.

3.8.1 Utilização:

- IconMoveFnc(handles,IconsSelected,indice_icon,a,b)

Parâmetros de entrada

handles: Estrutura que contém os handles dos objectos da janela do simulador OSIP.

IconsSelected Array que contém a posição, na estrutura *icons_schematic*, dos ícones seleccionados.

a Deslocamento segundo x.

b Deslocamento segundo y.

Parâmetros de saída:

Não existem.

Variáveis globais:

icons_schematic

Ver também: *mouseclickup, mouseclickdouwn, redraw_conections, RedrawConectionsSelected*.

Help Matlab:
- 'Handles'
- 'get'

3.8.2 Descrição do algoritmo:

Inicialmente verifica-se se o objectivo é mover um ícone ou um conjunto de ícones para o local dado pelo cursor, ou fazer um deslocamento dado pelas variáveis de entrada *a* e *b*.

Para o caso de se mover apenas um ícone, chama-se a função local *MoveFunc*. Esta função faz o deslocamento do ícone na área de desenho e actualiza a estrutura *icons_schematic* com os novos parâmetros do ícone. De seguida chama-se a função *conections* que determina quais os portos do ícone que se encontram ligados, fazendo o deslocamento da respectiva ligação através da função *redraw_conections*.

Para o caso de se pretender mover um conjunto de ícones previamente seleccionados. O funcionamento é igual ao caso de se pretender mover apenas um ícone, apenas com a diferença que é efectuado o deslocamento para todos os ícones seleccionados (a função *MoveFunc* é chamada para todos os ícones seleccionados).

Para fazer o deslocamento das ligações é chamada a função *conections* que determinar os portos que se encontram ligados e esta chama a função *RedrawConectionsSelected* que faz o deslocamento da ligação tendo em conta os troços da ligação que se encontram dentro da área de selecção.

O deslocamento dos ícones no esquemático é determinado tendo como referencia as coordenadas do cursor no esquemático. No entanto é possível efectuar um deslocamento que não tenha como referencia o cursor, para tal deve-se introduzir nos parâmetros de entrada *a* e *b* na função IconMoveFnc com o deslocamento pretendido segundo *x* e *y* respectivamente.

3.8.3 Fluxograma da função IconMoveFnc

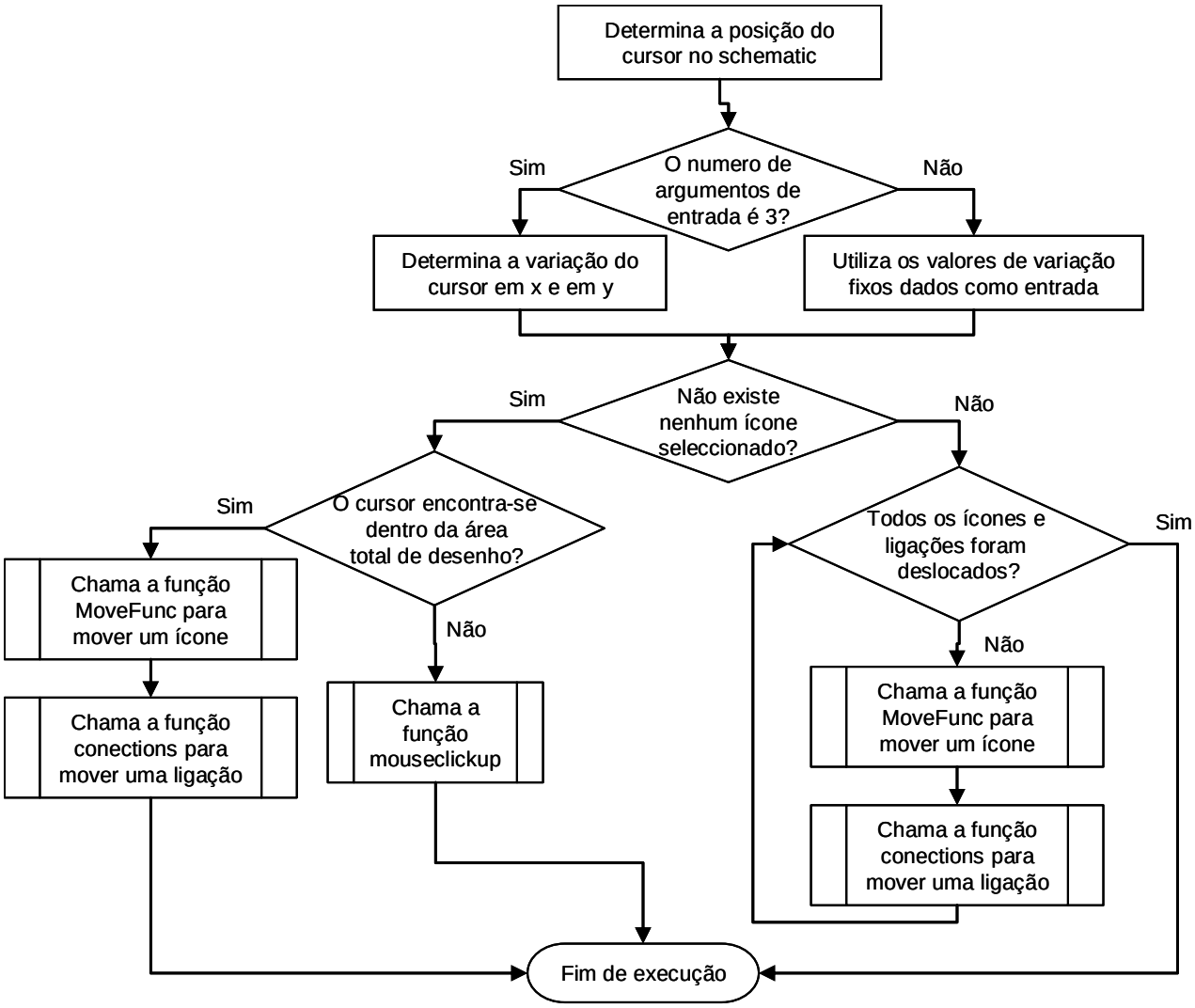


Figura 3.8.1 Fluxograma da função IconMoveFnc

3.8.4 Fluxograma da função local MoveFunc

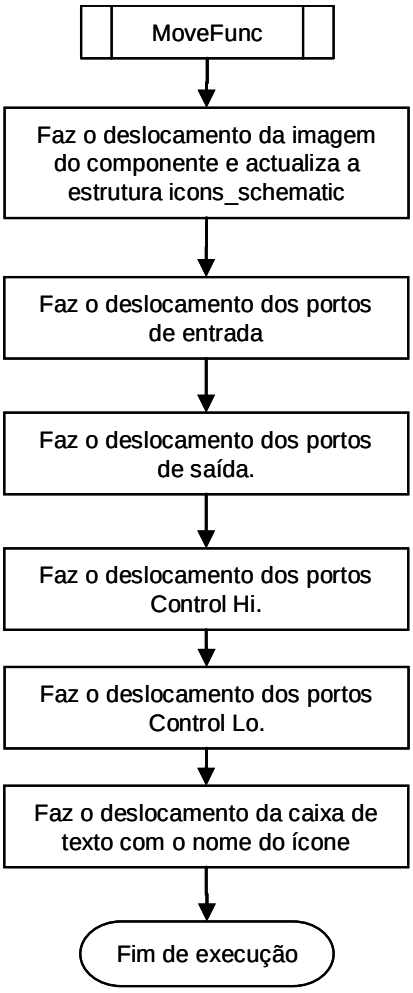


Figura 3.8.2 Fluxograma da função local MoveFunc

3.8.5 Fluxograma da função local conections

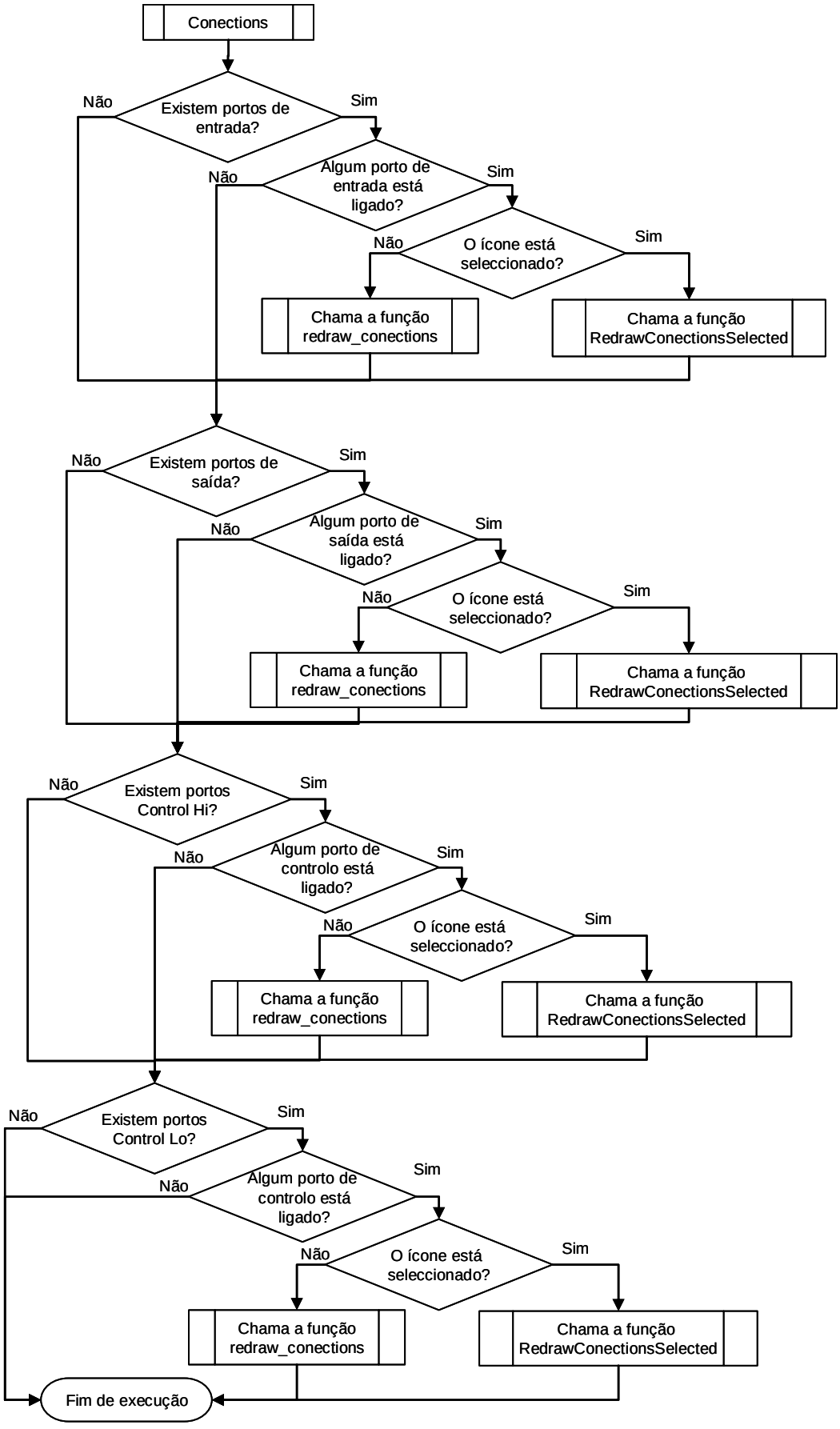


Figura 3.8.3 Fluxograma da função local conections

3.9 Função *select*

Esta função determina a área seleccionada com o rato.

A área seleccionada e devolvida na variável global *PositionSelected*, esta variável é um *array* de quatro posições, onde na primeira posição contém o valor inferior da área seleccionada em *x*, na 2ª posição o valor superior da área seleccionada em *x*, na 3ª posição o valor inferior da área seleccionada em *y* e na 4ª posição o valor superior da área seleccionada em *y*.

3.9.1 Utilização:

- select(handles)

Parâmetros de entrada:

handles: Estrutura que contém os handles dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

flag_select

BeginSelect

PositionSelected

Ver também: *mouseclickup, mouseclickdouwn*

Help Matlab:

– 'Handles'

– 'Line'

3.9.2 Descrição do algoritmo:

Em primeiro lugar determina-se as coordenadas do cursor na área de desenho. De seguida verifica-se se a *flag* de selecção se encontra desactivada (flag_select==0). Em caso afirmativo desenha um rectângulo na área de desenho com um tamanho dado pelas coordenadas do início da selecção (*BeginSelect*) e pelas coordenadas actuais do cursor.

Caso a *flag* de selecção se encontre activa, é eliminada a área de selecção anterior e redesenhada uma nova que tem início nas coordenadas indicadas pela variável *BeginSelect* e termina nas novas coordenadas do cursor.

Por fim são determinados os limites da área de selecção do valor inferior para o valor superior em *x* e em *y* e guardados na variável global *PositionSelected* nesta ordem.

3.10 Função redraw_conections

Esta função faz o deslocamento das ligações de um componente quando este é movido na área de desenho. O novo valor das ligações é actualizado na estrutura *ligacoes_icons*.

A posição do ícone na estrutura *ligacoes_icons* é dado como parâmetro de entrada assim como o porto na qual se encontra a ligação que se pretende deslocar, o tipo de porto (*in*, *out*, *control_H*, *control_L*), a estrutura *icons_schematic* que contém todos os parâmetros do ícone na área de desenho, a variação do deslocamento da ligação segundo *x* e a variação do deslocamento da ligação segundo *y*.

3.10.1 Utilização:

- [icons_schematic ligacoes_icons] = redraw_conections (icons_schematic,ligacoes_icons,type_port,deltax, deltay,indice_icon,index_port)

Parâmetros de entrada:

icons_schematic:	Estrutura que contém todas as características de um ícone na área de desenho.
ligacoes_icons:	Estrutura que contém as ligações dos ícones existentes na área de desenho.

type_port:	Qual o tipo de porto que se pretende alterar a ligação.
'in'	Para um porto de entrada.
'out'	Para um porto de saída.
'control_H'	Para um porto de controlo superior.
'control_L'	Para um porto de controlo inferior.
indice_icon	Índice do ícone na estrutura *icons_schematic* ou *ligacoes_icons* (a posição é a mesma na duas estruturas).
index_port	Índice do porto.
deltax	Deslocamento segundo *x*.
deltay	Deslocamento segundo *y*.

Parâmetros de saída:

icons_schematic:	Estrutura que contém todas as características dos ícones na área de desenho.
ligacoes_icons:	Estrutura que contém as ligações dos ícones na área de desenho.

Variáveis globais:

flag_select	Flag que indica quando se encontra a 1 que foi seleccionada uma área no schematic.

Ver também: *LigarIcon, conect_port*.

Help Matlab:
- 'Set'
- 'Line'

3.10.2 Descrição do algoritmo:

Em primeiro lugar começa-se por verificar se a ligação que pretende deslocar é de um porto de entrada. Em caso afirmativo verifica se esta está terminada, se estiver,

determina qual o ícone de destino da ligação através do comando dest=ligacoes_icons(indice_icon).porto_in(index_port).dest e qual o índice do porto através do comando i=round((ligacoes_icons(indice_icon). porto_in(index_port).y(end)-(icons_schematic(dest).posicao.y-25))* (icons_schematic(dest).n_portos_out+1)/50). Adverte-se que os portos são desenhados de uma forma automática para mais informações ver a função *mouseclickup*.

A seguir verifica se a ligação que se pretende deslocar é constituída por mais do que um troço. Em caso afirmativo, determina a posição do troço que se encontra ligado directamente ao porto, determina a nova posição do troço, tendo em conta o deslocamento *deltax* e *deltay* (parâmetros de entrada) que se pretende efectuar. O deslocamento da ligação é efectuado através do comando set(ligacoes_icons(indice_icon).porto_in(index_port).y(2),'xdata',x,'ydata',y).

Se a ligação for constituída por um único troço, ligação directa entre os portos, é apagada a ligação existente e desenhada uma nova tendo como referência a posição dos portos na área de desenho.

Por fim é actualizada a estrutura *ligacoes_icons* com os novos parâmetros da ligação.

A explicação anterior é para o caso de se pretender deslocar uma ligação de um porto de entrada. No entanto o raciocínio aplica-se para os restantes portos (*port_out*, *control_H* e *control_L*).

3.10.3 Fluxograma da função redraw_conections

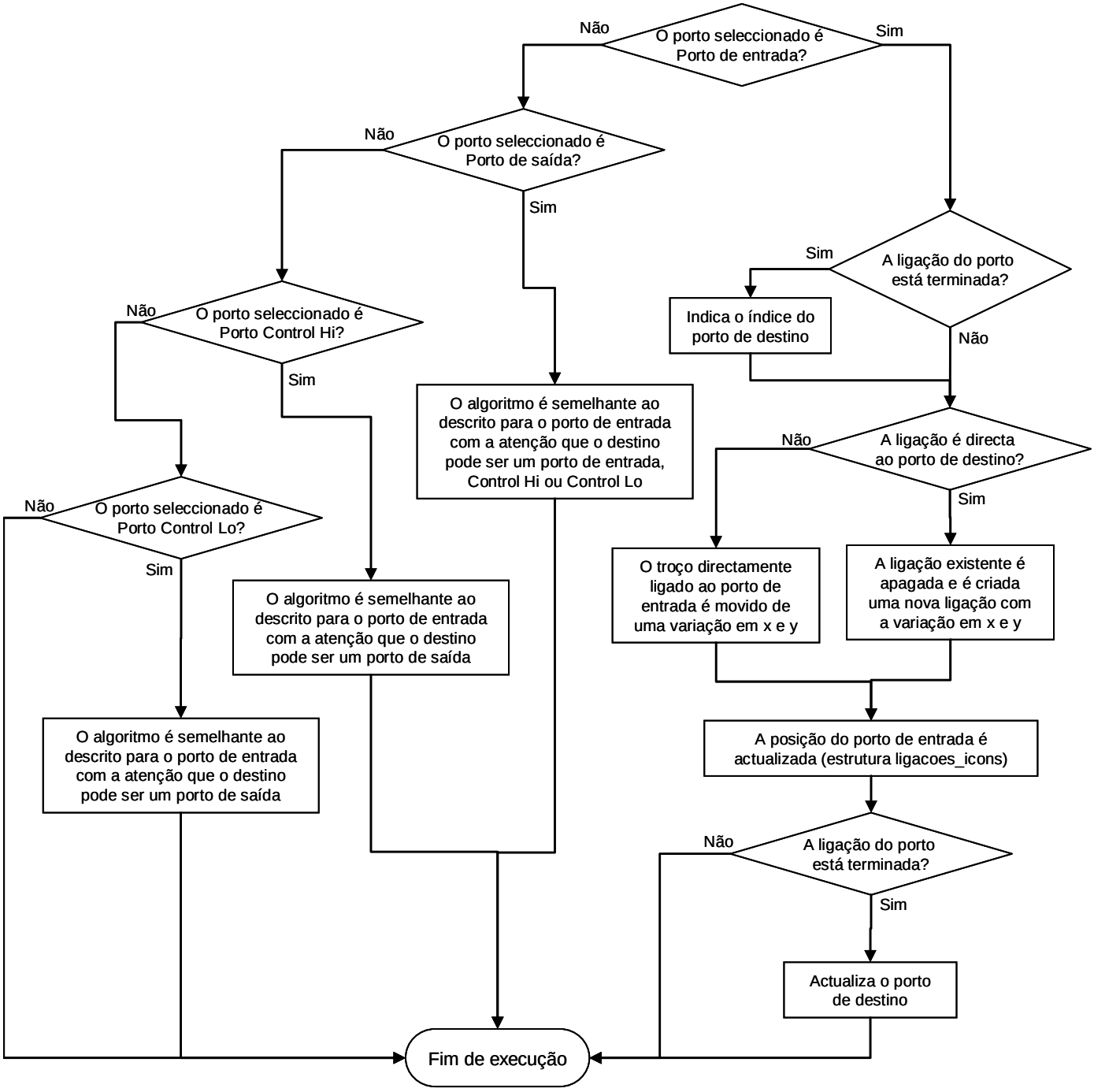


Figura 3.10 Fluxograma da função redraw_conections

3.11 Função *RedrawConectionsSelected*

Esta função faz o deslocamento de uma ligação de um ícone quando se move um conjunto de ícones no esquemático, previamente seleccionados.

Esta função é chamada pela função *conections* dentro da função *IconMoveFnc*.

3.11.1 Utilização:

- [icons_schematic ligacoes_icons]= RedrawConectionsSelected(icons_schematic, ligacoes_icons, type_port, deltax, deltay, indice_icon, index_port)

Parâmetros de entrada:

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
ligacoes_icons	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.
type_port	Tipo do porto da ligação que se pretende deslocar.
deltax	Deslocamento da ligação segundo *x*.
deltay	Deslocamento da ligação segundo *y*.
icon_index	Índice do ícone na estrutura *icons_schematic*.
index_port	Índice do porto.

Parâmetros de saída:

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
ligacoes_icons	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.

Variáveis globais:

IconsSelected	Array que contém os ícones seleccionados na área de desenho.
PositionSelected	Array com os limites da área seleccionada na área de desenho.
HandlesConectSelec:	Estrutura que contém os handles dos troços das ligações que se encontram dentro da área de selecção, para ligações que se iniciam dentro da área de selecção mas terminam fora.
HandlesConectSelec(i).in(j).handle	Handles dos troços das ligações dos portos de entrada '*in*'.
HandlesConectSelec(i).out(j).handle	Handles dos troços das ligações dos portos de saída '*out*'.

HandlesConectSelec(i).control_H(j).handle	Handles dos troços das ligações dos portos de controlo superior '*control_H*'.
HandlesConectSelec(i).control_L(j).handle	Handles dos troços das ligações dos portos de controlo inferior '*control_L*'.

Nota: 'i' indica o índice do ícone e 'j' o índice do porto

Ver também: *mouseclickup, mouseclickdown*, *IconMoveFnc*, *HandlesConections*.

Help Matlab:

- 'Set'
- 'Get'
- 'Line'

3.11.2 Descrição do algoritmo:

A descrição feita neste ponto é referente a uma ligação entre um porto de entrada '*in*' e um porto de saída '*out*'. Para os restantes casos a metodologia aplicada é a mesma.

Inicialmente o algoritmo começa por verificar qual o tipo do porto, considerando que se trata de um porto de entrada, verifica se a ligação se encontra terminada, ou seja se existe um destino da ligação. Caso a ligação se encontre terminada determina-se qual o ícone e o porto de destino da ligação.

De seguida verifica-se se o ícone de destino da ligação se encontra dentro da área de selecção, em caso afirmativo é efectuado o deslocamento da ligação, uma ligação pode ser constituída por um único troço ou por um conjunto de troços.

No caso do ícone de destino da ligação se encontrar fora da área de selecção apenas são deslocados os troços da ligação que se encontram dentro da área de selecção. Estes troços são obtidos através da estrutura *HandlesConectSelec*. Esta estrutura contém os handles dos troços da ligação que se encontram dentro da área de selecção, acedendo à seguinte posição da estrutura: HandlesConectSelec(indice_icon).in(index_port).handle, onde

'indice_icon' indica o índice do ícone e 'index_port' o porto onda a ligação se encontra estabelecida.

A ligação no porto do ícone de destino também é actualizada com as novas posições da ligação (estrutura *ligacoes_icons*).

Se por algum motivo a ligação não se encontrar terminada, é feito o deslocamento de todos os troços da ligação, mesmo os troços da ligação que se encontram fora da área de selecção.

A discrição feita anteriormente aplica-se para todo o tipo de portos, com a diferença da posição dos handles dos troços das ligações na estrutura *HandlesConectSelec* que depende do tipo de porto.

3.11.3 Fluxograma da função RedrawConectionsSelected

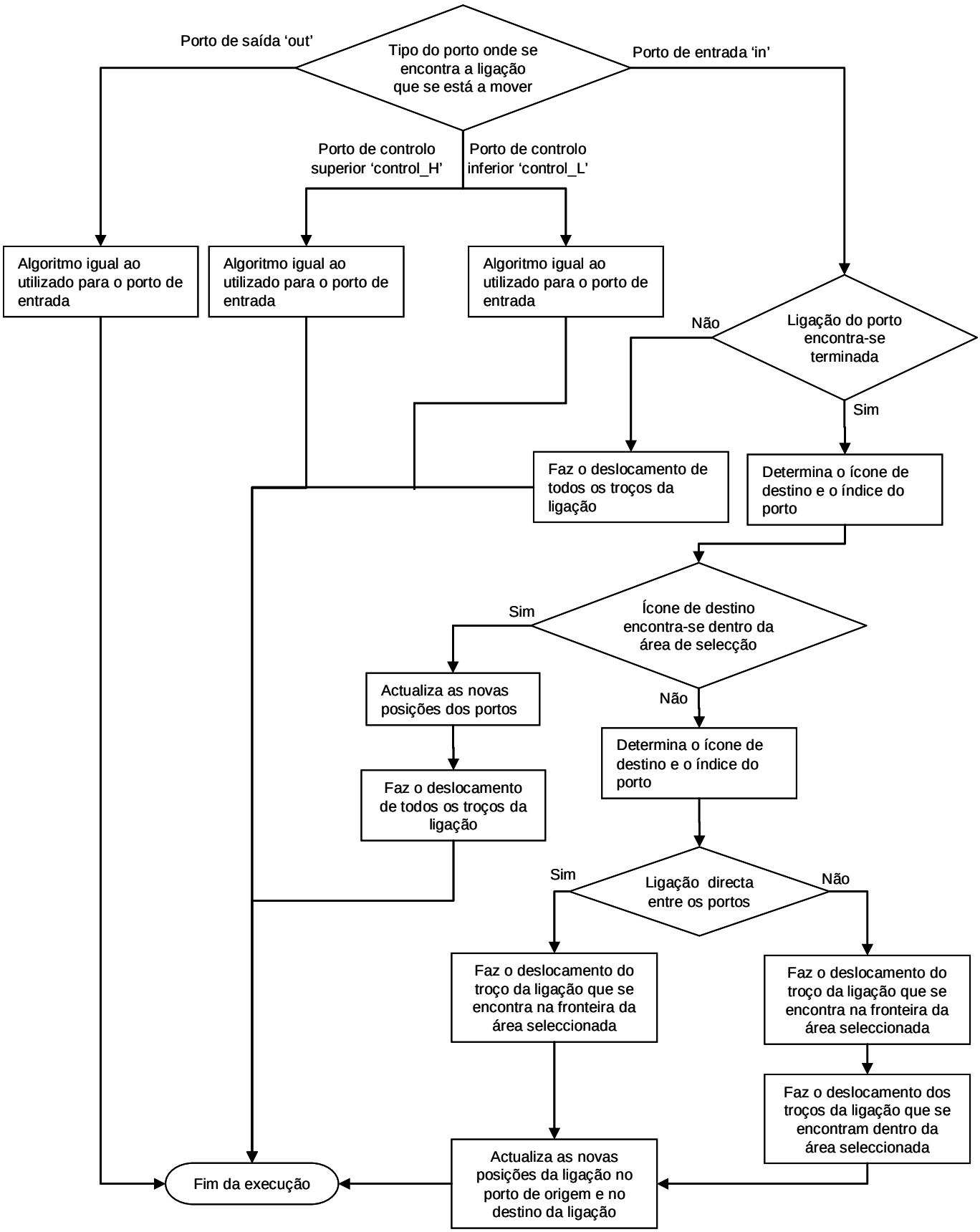


Figura 3.11 Fluxograma da função RedrawConectionsSelected

3.12 Função *keypressfunc*

Esta função é executada quando se pressiona uma tecla do teclado (Teclas usadas nesta versão – *Delete* e *Escape*).

Quando se pressiona a tecla D*elete* estando o cursor sobre um ícone, este é apagado assim como toda as ligações a este ícone.

Caso o cursor se encontre sobre um porto, é apenas apagada a ligação desse porto.

Se for pressionada a tecla *Delete* depois de seleccionar um conjunto de ícones na área de desenho, todos os ícones seleccionados são apagados e as respectivas ligações que estes contêm.

No caso de se pressionar a tecla *Escape* quando se está a efectuar uma ligação, esta é interrompida voltando-se ao funcionamento normal.

Se a tecla *Esc*ape for pressionada quando se tem o zoom activado, este é desactivado e volta-se ao funcionamento normal.

O teste para verificar qual a tecla pressionada é efectuada comparando com o *ASCII* do teclado. Os ASCII das teclas referidas anteriormente são: 27 para a tecla *Escape* e 127 para a tecla *Delete*.

3.12.1 Utilização:

- keypressfunc(handles)

Parâmetros de entrada:

handles: Estrutura que contém os handles dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

flag_liga

icon_liga

icons_schematic

ligacoes_icons

nicons

icons_valid

flag_select

IconsSelected

flag_ZoomOut

flag_ZoomIn

icon_posi

und

Ver também: *delete_connection, DeleteIcon*

Help Matlab:

- 'Set'
- 'Get'
- 'CurrentCharacter'

3.12.2 Descrição do algoritmo:

Inicialmente determina-se a posição da janela OSIP no ecrã, as coordenadas do cursor, a posição da área de desenho na janela OSIP e os limites em *x* e em *y* da área de desenho.

De seguida determina-se qual a tecla do teclado pressionada através do comando: get(handles.osip_f,'CurrentCharacter') e determina-se o *ASCII* correspondente.

Se for pressionada a tecla *Escape* as flags *flag_liga*, *flag_ZoomOut* e *flag_ZoomIn* são colocadas a zero.

Depois determina-se qual a forma do cursor através do comando: get(handles.osip_f,'Pointer').

Se for pressionada a tecla *delete* e a forma do cursor for um círculo, quer dizer que o cursor se encontra sobre um porto, como tal é eliminada a ligação existente neste porto através da função *delete_connection*. É ainda salvaguardado o esquemático antes da eliminação da ligação, para posterior retrocesso, chamando a função *undo_function.m*.

Uma outra situação é pressionar a tecla *delete* quando o cursor se encontra dentro da área de desenho com a forma de dupla seta em cruz '*fleur*', nesta situação o cursor encontra-se sobre um ícone na área de desenho, como tal é eliminado o ícone da área de desenho através da função local *DeleteIcon*. Esta função verifica se o ícone que se pretende eliminar se encontra na variável *Paste_icon*, variável que contem o índice do ícone copiado, em caso afirmativo a variável *Paste_icon* é colocada a vazio (*Paste_icon*=[]) e é desactivada a função de *Paste*. De seguida são apagadas as ligações dos portos do ícone, chamando para cada porto a função *delete_connection*, apagada a imagem do componente, decrementado o número de ícones na área de desenho e eliminados os parâmetros do ícone na estrutura *icons_schematic*. É ainda guardado o esquemático antes de apagar o ícone, para a possibilidade de posterior retrocesso, através da função *undo_function.m*.

Por fim se for pressionado a tecla *delete* estando a flag de selecção activa flag_select==1, são eliminados todos os ícones que se encontram dentro da área de selecção, para tal é chamada a função *DeleteIcon* para cada ícone existente na área de selecção e de seguida actualizada a variável que contém os índices dos ícones que se encontram dentro da área de selecção, por ultimo coloca-se a flag flag_select==0. Mais uma vez é ainda salvaguardado o esquemático antes da eliminação dos ícones seleccionados, para posterior retrocesso, chamando a função *undo_function.m*.

3.12.3 Fluxograma da função keypressfunc

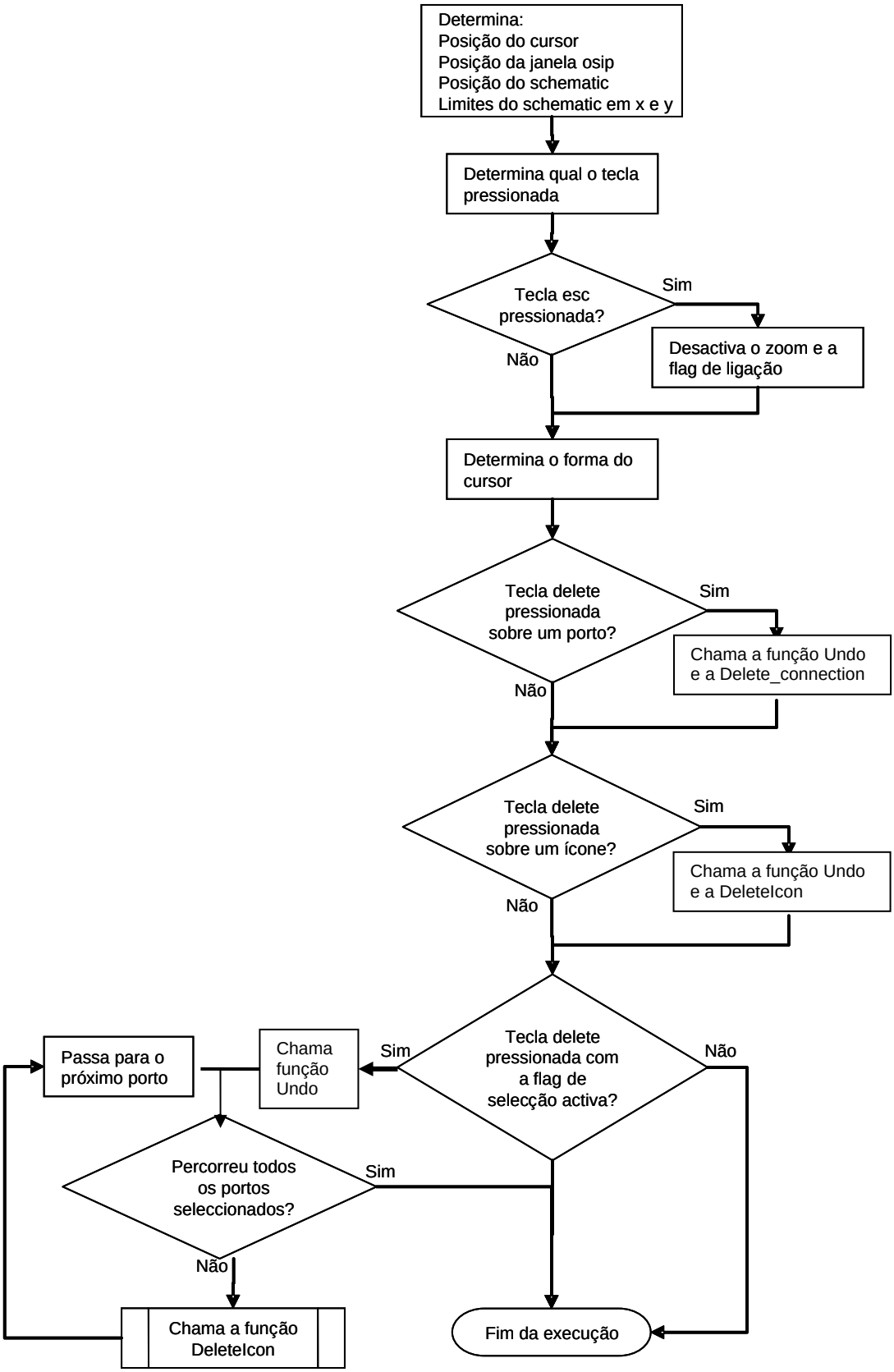


Figura 3.12.1 Fluxograma da função keypressfunc

3.12.4 Fluxograma da função local DeleteIcon

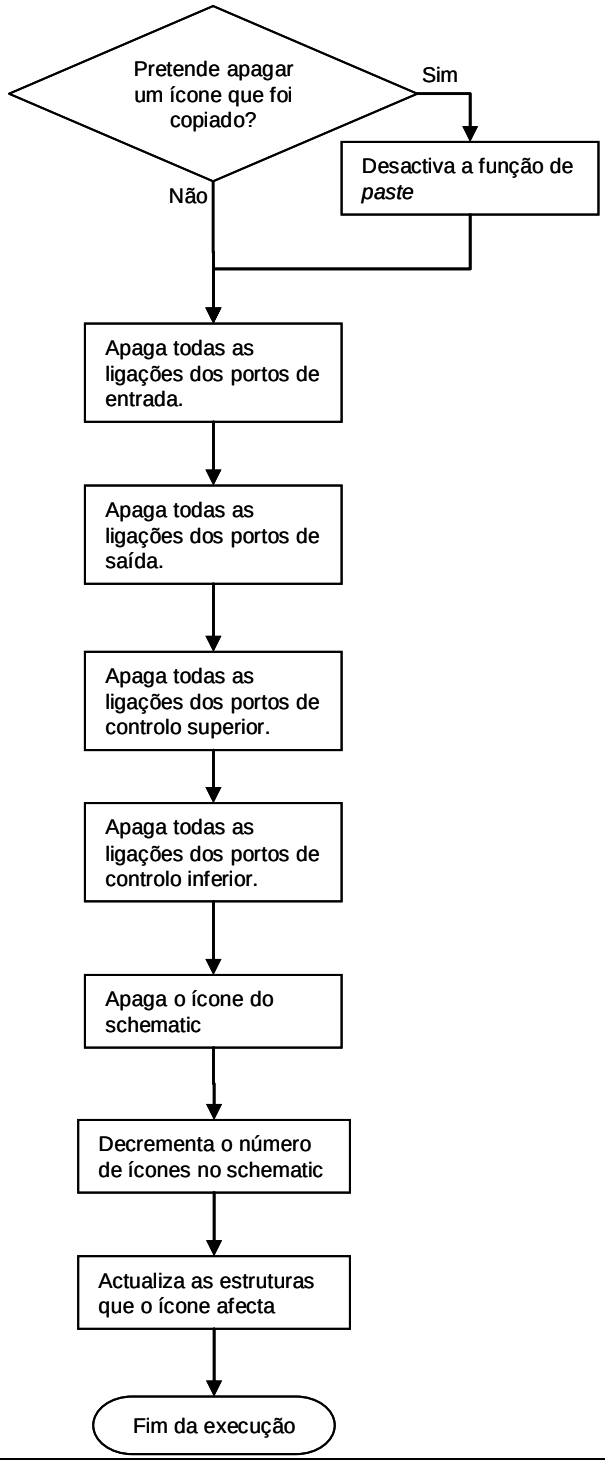


Figura 3.12.2 Fluxograma da função DeleteIcon

3.13 Função *conect_port*

Esta função é chamada quando se está a efectuar uma ligação entre dois portos. A função é chamada automaticamente quando se move o cursor com o botão esquerdo

do rato pressionado, a partir do momento que se iniciou a ligação ou seja a partir do momento que se faz um click num porto de um ícone presente na área de desenho. Os vários troços das ligações vão sendo guardados na estrutura *ligacoes_icons* nas posições que dependem do índice do ícone, do tipo de porto e do número de troços da ligação.

3.13.1 Utilização:

- conect_port(handles)

Parâmetros de entrada

handles	Estrutura que contém os handles dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis Globais:

nicons

icons_schematic

ligacoes_icons

icons_valid

icon_liga

Ver também: *OSIP, osip_f, mouseclickdown*

Help Matlab:

– 'handles'

– 'Set'

– 'Get'

– 'Line'

3.13.2 Descrição do algoritmo:

Inicialmente é determinada a posição da janela do simulador no ecrã, as coordenadas do cursor no ecrã, a posição da área de desenho dentro da janela do simulador e os limites da área de desenho em x e y.

Depois verifica-se se o cursor se encontra sobre um porto de entrada (através do campo *type* na estrutura *icon_liga*). Em caso afirmativo verifica se ainda não foi iniciada uma ligação neste porto (através da estrutura *ligacoes_icons*). Em caso afirmativo, inicia-se a ligação, desenhando o troço até ao local onde de encontra o cursor. O porto é ainda alterado de um quadrado para uma seta que indica o sentido de propagação do sinal.

Para o caso de já se ter iniciado a ligação, sempre que se move o cursor do rato, apaga-se o troço e desenha-se um novo até ao novo local onde de encontra o cursor e adiciona o *handle* do troço à estrutura *ligacoes_icons*.

Para o caso da ligação já ter um ou mais troços terminados, o novo troço é iniciado nas coordenadas do último troço. Sempre que se move o cursor do rato, apaga o troço e desenha um novo até ao local actual do cursor e o *handle* do troço é adicionado à estrutura *ligacoes_icons.* Este processo repete-se até ser terminada a ligação no porto de destino ou se interromper a ligação pressionando a tecla *Escape*.

Para os portos de saída e de controlo o processo é semelhante ao descrito para o porto de entrada com as devidas adaptações nas estruturas e variáveis utilizadas.

3.13.3 Fluxograma da função Conect_port

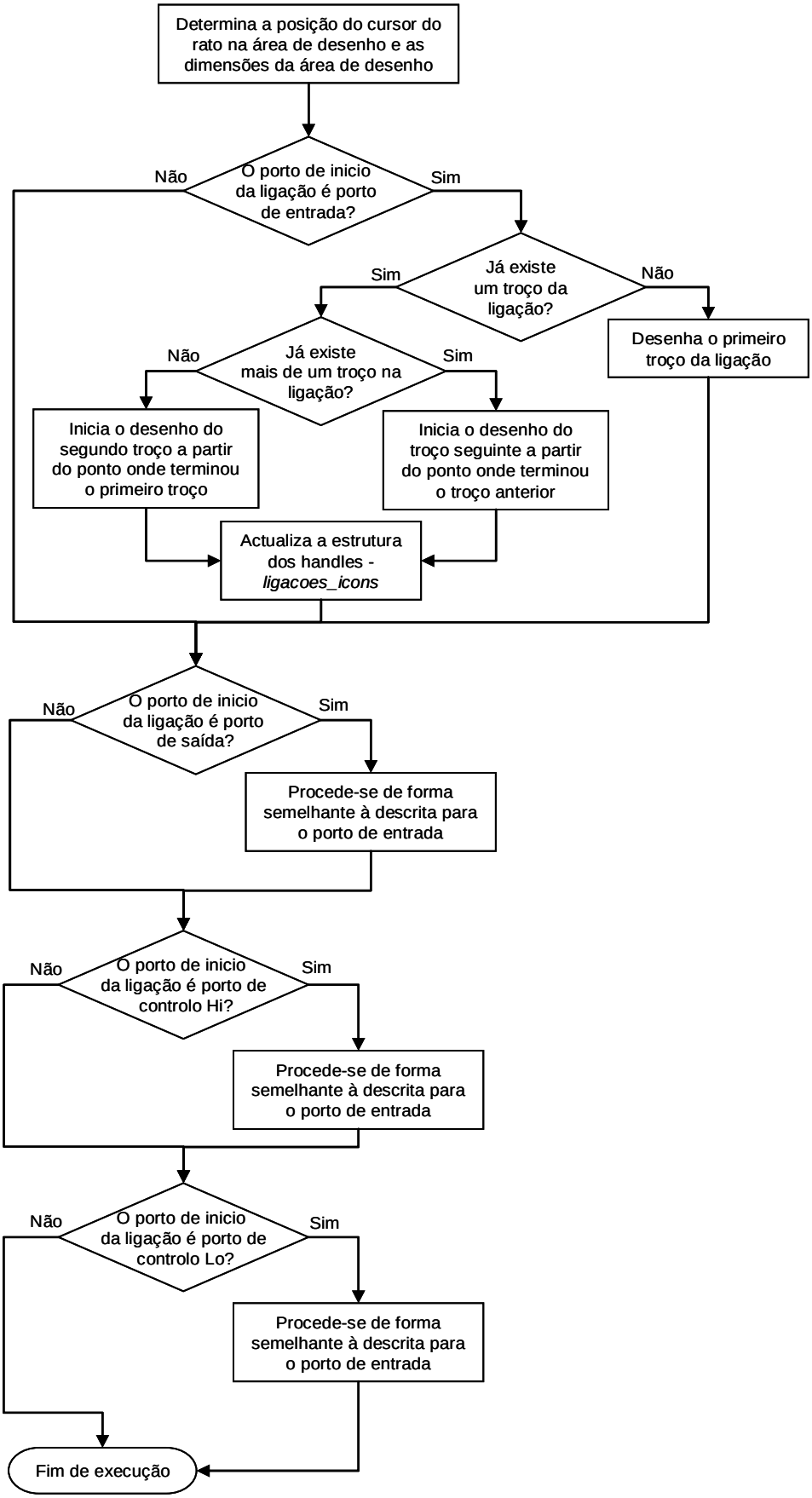


Figura 2.1.13.1 Fluxograma da função Conect_port

3.14 Função *delete_connection*

Esta função apaga uma ligação entre dois portos quando se pressiona a tecla *delete* e o cursor se encontra sobre um porto. A forma do cursor quando este se encontra sobre um porto é uma circunferência.

Esta função também é chamada quando se elimina um componente da área de desenho, caso este se encontre ligado para eliminar as ligações.

3.14.1 Utilização:

- delete_connection(type,posi,porto)

Parâmetros de entrada

Tipo de porto	– in – out – control_H – control_L	Indica o tipo do porto onde se pretende eliminar a ligação.
posi	Indica a posição do ícone na estrutura *ligacoes_icons* ou *icons_schematic*.	
Porto	Indica o índice do porto.	

Parâmetros de saída:

Não existem

Variáveis Globais:

icons_schematic

ligacoes_icons

Ver também: *OSIP* e *keypressfunc*

Help Matlab:

– 'handles'

– 'delete'

– 'Set'

3.14.2 Descrição do algoritmo:

Inicialmente verifica se o porto onde o cursor se encontra para eliminar a ligação é um porto de entrada (*type = 'in'*). Em caso afirmativo, verifica se o porto tem uma ligação efectuada (terminada ou não). Caso não tenha imprime uma mensagem de erro e termina a execução deste programa. Se a ligação existir, então vai verificar se a ligação está terminada, em caso afirmativo determina o porto de destino da ligação (neste caso é um porto de saída). De seguida elimina as entradas na estrutura *ligacoes_icons* para o porto seleccionado e para o porto de destino, por ultimo apaga-se a ligação.

Caso a ligação não esteja terminada então são eliminadas as entradas na estrutura *ligacoes_icons* para o porto seleccionado e apaga-se a ligação.

Para os restantes tipos de portos a descrição é semelhante, tendo-se apenas que ter em atenção que para os portos do tipo '*out*' podem ter ligações a portos de entrada, a portos Control Hi ou a portos Control Lo. Assim é necessário testar o tipo de porto de destino para criar o porto, ao se eliminar a ligação.

3.14.3 Fluxograma da função delete_connection

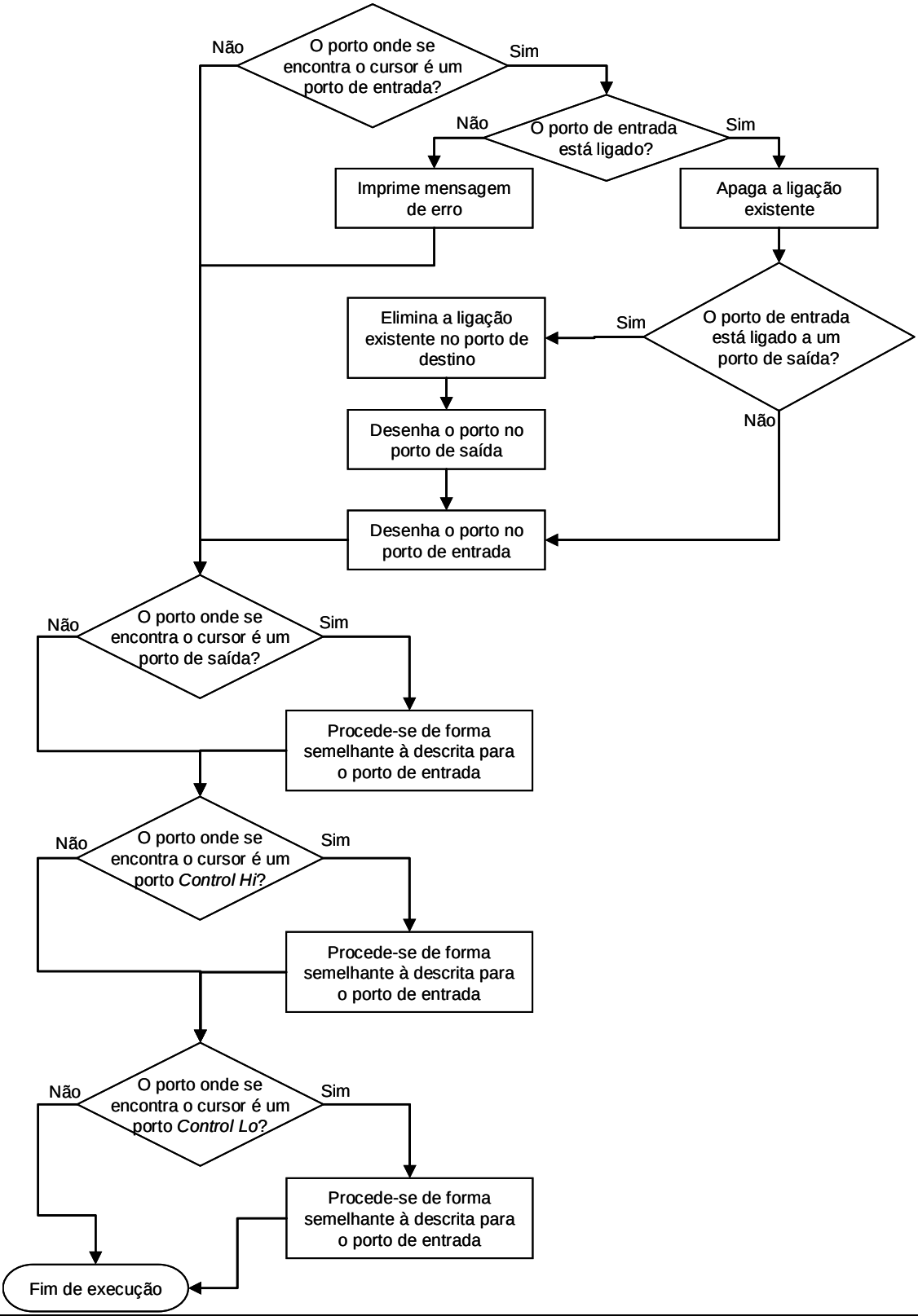


Figura 3.14 Fluxograma da função delete_connection

3.15 Função *CopyFunc*

Esta função copia um ícone da área de desenho e apenas um.

Existem dois modos de copiar um ícone da área de desenho. O método mais rápido é clicar com o botão direito do rato no ícone que se pretende copiar e de seguida escolher a opção '*Copy*'. O segundo método consiste em seleccionar apenas um ícone na área de desenho, ir ao menu '*Edit*' e seleccionar a opção 'Copy'.

3.15.1 Utilização:

- CopyFunc(handles)

Parâmetros de entrada:

handles: Estrutura que contém os handles dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

paste_icon

IconsSelected

icon_posi

Ver também: *OSIP* e *PasteFunc*

Help Matlab:

– 'handles'

– 'delete'

– 'Set'

3.15.2 Descrição do algoritmo:

Inicialmente verifica-se se não existem ícones seleccionados na área de desenho, através da variável *IconsSelected*, em caso afirmativo é guardado na variável *paste_icon* o índice do ícone sobre o qual o cursor se encontra, que é dado pela variável *icon_posi*.

Caso existam ícones seleccionados (para o caso de se seleccionar apenas um ícone) é guardado na variável *paste_icon* o índice do ícone seleccionado que se encontra na variável *IconsSelected*, eliminada a selecção existente e desactivada a função de *Copy* no menu *Edit* do *OSIP*.

Depois de copiar o índice do ícone, activa-se a funções de *Paste* do menu *Edit* do *OSIP* através do seguinte comando set(handles.paste,'Enable','on') e a função de *Paste* quando se clica com o botão direito do rato dentro na área de desenho, através dos seguintes comandos:

```
Children_cmenu=get(handles.chematic_cmenu,'Children')
set(Children_cmenu(2),'Enable','on')
```

3.16 Função *PasteFunc*

Esta função faz o paste de um ícone que se foi copiado previamente na área de desenho.

Esta função é que se encarrega de colocar um novo ícone na área de desenho e actualizar todas as estruturas com os parâmetros do novo ícone. Os parâmetros do novo ícone são exactamente iguais às do ícone copiado com a excepção da posição do ícone.

3.16.1 Utilização:

- PasteFunc(handles)

Parâmetros de entrada:

handles: Estrutura que contém os handles dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

paste_icon

icons_schematic

nicons

icons_valid

IconsSelected

ligacoes_icons

flag_select

BeginSelect

PositionSelected

Ver também: *OSIP*, *CopyFunc, mouseclickup*

Help Matlab:

– 'uicontextmenu'

– 'image'

– 'uimenu'

3.16.2 Descrição do algoritmo:

Começa-se por determinar qual o nome do componente correspondente ao ícone copiado através do campo *nome* da estrutura *icons_schematic* na posição dada pela variável *paste_icon*. Esta variável contém o índice do ícone que se pretende colar.

De seguida determina-se a posição na estrutura *icons_schematic* onde colocar o novo ícone.

Posteriormente guarda-se para uma variável local os parâmetros do ícone que se copiou através do seguinte comando: Icon=icons_schematic(paste_icon), onde *paste_icon* contém o índice do ícone que se pretende duplicar.

Depois utiliza-se uma variável local (*Ligacoes*) para inicializar a estrutura das ligações para todos os portos do novo ícone. Incrementa-se o número total de ícones existentes na área de desenho e o número de ícones do tipo de componente adicionado.

A seguir determina as coordenadas *x* e *y* da área de desenho onde colocar o novo ícone. Se for utilizado o menu *Edit* do OSIP, length(IconsSelected)==1, para se efectuar o paste, as coordenadas são determinadas com base no ícone que se copiou. Caso seja

utilizado o botão direito do rato, utiliza-se as coordenadas do rato no momento que se faz a operação de colar.

Sabendo as coordenadas, coloca-se a imagem do componente correspondente e desenha-se todos os portos guardando as posições nos campos da estrutura *Icon*. Actualiza-se na estrutura *Icon* o campo *nome* com o nome do novo ícone, guarda-se os *handles* do novo ícone, a sua posição na área de desenho e coloca a prioridade com o valor por defeito '-1'.

Activa as funções de *copy* no novo ícone e copia-se os parâmetros do componente para os novos ficheiros dado pelo nome do novo ícone na área de desenho.

Por fim atribui o valor das estruturas *Icon* à estrutura *icons_schematic* na posição dada pelo índice do novo ícone através do seguinte comando: icons_schematic(n)=Icon, onde *n* é o índice do novo ícone na estrutura *icons_schematic*. Também se atribui o valor da variável local *Ligacoes* estrutura *ligacoes_icons* na posição do novo ícone através do seguinte comando: Ligacoes_icons(n)=Ligacoes e activa-se a nova posições nas estruturas colocando no array *icons_valid*, na posição correspondente ao novo ícone, o valor '1' (icons_valid(n)=1).

3.16.3 *Fluxograma da função PasteFunc*

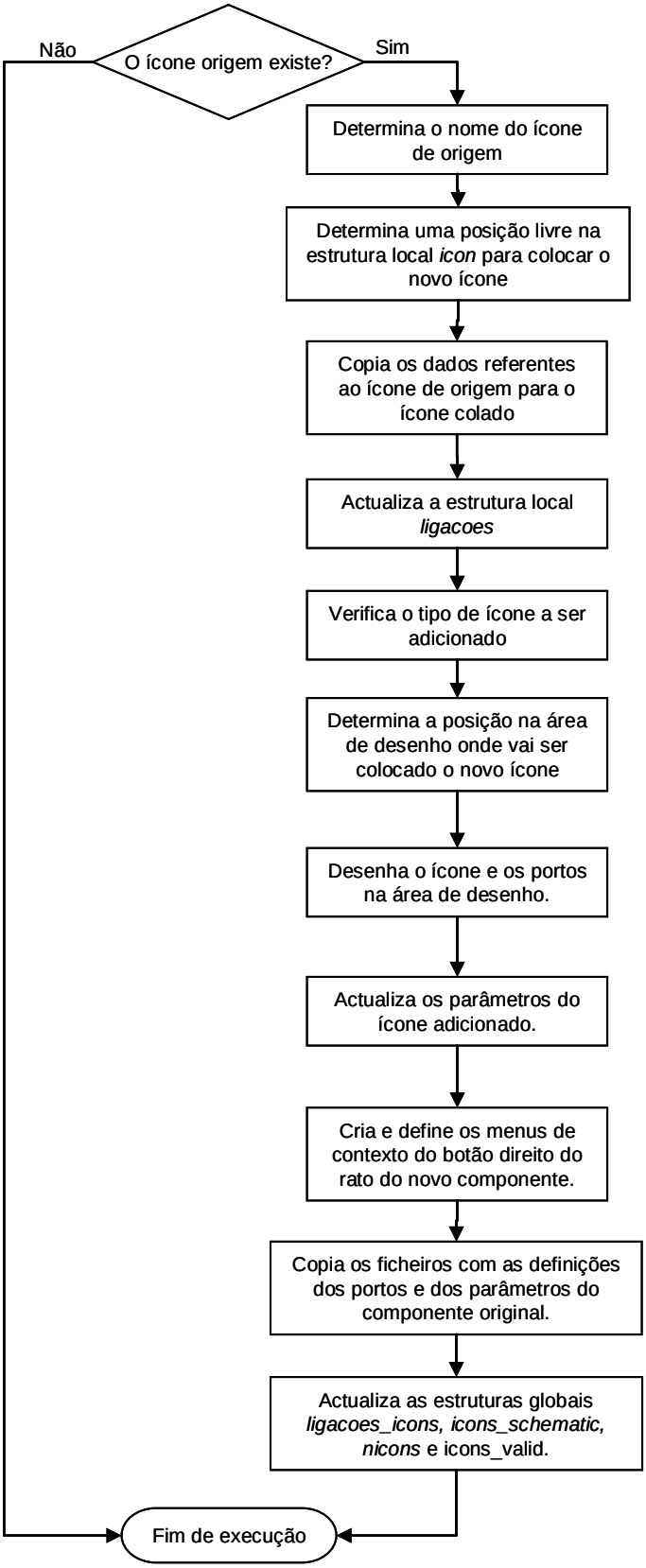


Figura 3.16 Fluxograma da função Past_function

3.17 Função *Run_Simulation*

Esta função efectua a simulação do circuito presente na área de desenho, começando por verificar se todos os componentes se encontram ligados. Se existir algum porto que se encontre desligado é interrompida a simulação e enviada uma mensagem de erro ao utilizador.

Se todos os portos dos ícones se encontrarem ligados, passa-se para a fase de atribuição de prioridades, para tal é escolhido um ícone de início de sequência e atribuem-se as prioridades a todos os ícones desse ramo.

3.17.1 Utilização:

- icons_schematic=run_simulation(icons_schematic, ligacoes_icons,icons_valid,NUM)

Parâmetros de entrada

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
ligacoes_icons	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.
icons_valid	Array que indica quando a '1' qual a posição na estrutura *icons_schematic* é válida.
NUM	Estrutura que define os parâmetros genéricos.

Parâmetros de saída:

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.

Variáveis Globais:

Não existem.

Ver também: *OSIP*, *branch_route* e *write_mfile*

Help Matlab:

– 'handles'

– 'rehash'

3.17.2 Descrição do algoritmo:

Inicialmente inicializa a *flag*: flag_error=0; e verifica se já existe algum ficheiro de execução da simulação criado. Se existir apaga-o, senão testa os portos dos ícones existentes na área de desenho a nível das ligações. Se algum porto (de qualquer tipo) não estiver ligado, então sinaliza o primeiro porto encontrado nestas condições e termina a execução do programa enviando uma mensagem de erro.

Para o caso de todos os portos se encontrarem ligados, então cria e abre o ficheiro de execução da simulação pelo comando: fid=fopen('..\temp\run_s.m','a'), verifica se a abertura foi bem sucedida e determina o número de ícones na área de desenho.

Inicia a escrita do ficheiro de execução da simulação escrevendo o cabeçalho e inicializando a estrutura *sinal* que será utilizada para fazer a passagem dos sinais entre os diversos componentes. Escreve ainda o código referente à criação e inicialização da barra de espera que se executa durante a execução da simulação.

De seguida, começa por procurar ícones de início de execução. Estes ícones podem ser de dois tipos: ícones que não têm portos de entrada nem de controlo (só têm portos de saída), ou ícones cujos portos de entrada ou de controlo (Hi ou Lo) estão ligados a ícones, aos quais já foi atribuída uma prioridade.

Para o caso de não encontrar nenhum dos dois tipos referidos, é enviada a mensagem: errordlg('No starting point found','!!Error!!');.

Após encontrar um ícone de início de sequência, é chamada a função [priority,icons_schematic]=branch_route(icons_schematic,ligacoes_icons,icons_valid,i,priority), que é responsável por escrever a parte de código no ficheiro de execução da simulação de todo o ramo até encontrar um ícone que quebre o ramo (com dois ou mais portos de entrada/controlo ou então com dois ou mais portos de saída). Nesta situação volta-se a procurar novo ícone de início de sequência.

No final quando todos os ícones tiverem prioridade e estiverem escritos no ficheiro de execução da simulação, o programa termina a barra de espera da simulação e fecha este ficheiro.

Executa de seguida o refrescamento das m-files, estrutura e variáveis internas do Matlab e executa o ficheiro *Run_s.m* , terminando o programa.

3.17.3 Fluxograma da função Run_Simulation

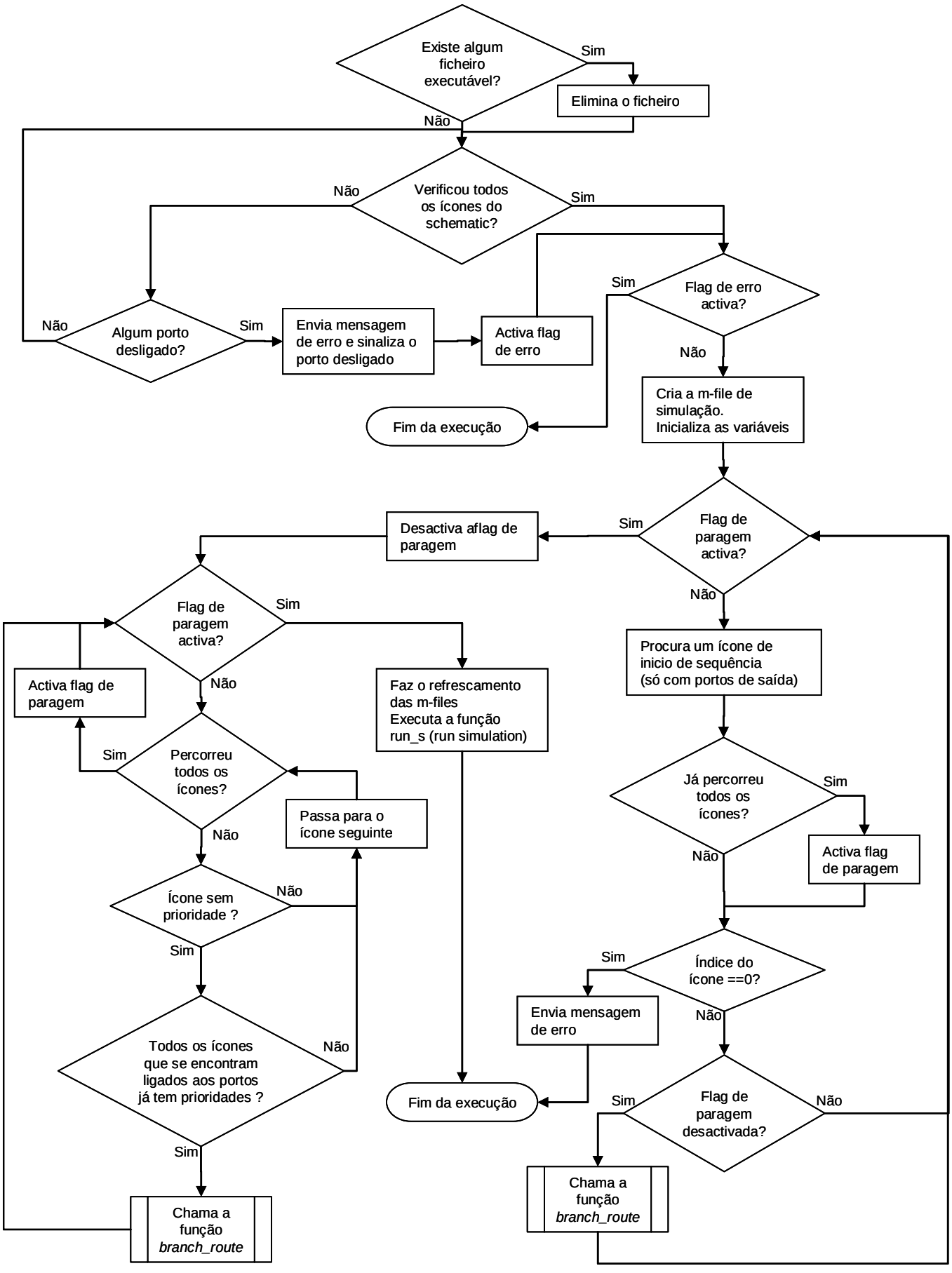


Figura 3.17 Fluxograma da função Run_Simulation

3.18 Função *Branch_Route*

Esta função é chamada para todos os ícones de início de sequência. Os ícones de início de sequência são ícones que não tem portos de entrada ou ícones que têm mais de um porto de entrada, saída ou contém portos de controlo.

Uma sequência é iniciada num ícone que respeita as condições anteriores. Define-se "sequência" como o conjunto de ícones que se encontram ligados no troço com apenas um porto de entrada e um porto de saída. O troço termina quando se encontra um ícone que não tem portos de saída ou um novo ícone de início de sequência.

Esta função atribui as prioridades aos ícones do troço.

Ao encontrar um ícone que termina o troço é chamada a função para escrever na m-file *run.m* as funções pela ordem correcta atribuindo os parâmetros de entrada a cada função.

Este processo repete-se para todos os ícones de início de sequência.

3.18.1 Utilização:

- [priority,icons_schematic]=branch_route(icons_schematic,ligacoes_icons,icons_valid,pos_index,priority);

Parâmetros de entrada

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
ligacoes_icons	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.
icons_valid	Array que indica quando a '1' qual a posição na estrutura *icons_schematic* é válida.
pos_index	Indica a posição do ícone na estrutura *icons_schematic*.
Prioridade	Valor da prioridade do último ícone.

Parâmetros de saída:

icons_schematic	Estrutura que contém informação dos ícones presentes

na área de desenho.

Prioridade Valor da prioridade do último ícone.

Variáveis Globais:

Não existem.

Ver também: *OSIP_f*, *run_simulation* e *write_mfile*

Help Matlab:

– 'handles'

– 'struct'

3.18.2 Descrição do algoritmo:

Inicialmente são declaradas e inicializadas as variáveis locais. A prioridade é incrementada e atribuída ao ícone actual. De seguida é verificado se o porto actual não tem portos de saída. Se não tiver, reinicializa as variáveis locais (*array_val* e *flag_begin*), atribui a posição actual do ícone na estrutura *icons_schematic* (*pos_index*) e escreve o código correspondente ao componente do ícone actual, no ficheiro de execução da simulação.

Se o ícone actual tiver portos de saída, então, para todos os portos de saída, guarda o índice do ícone de início de sequência e determina o ícone sucessor.

De seguida e até encontrar um ícone terminador (sem portos de saída) ou um ícone de início de sequência, se o ícone seguinte tiver 1 ou nenhum porto de saída, então incrementa a prioridade, atribui a prioridade ao ícone actual e adiciona a posição à estrutura *array_val* (indica a sequencia dos ícones no troco).

No caso do ícone ter um porto de saída, então passa para o ícone seguinte. Caso contrário, chama a função icons_schematic=write_mfile(icons_schematic,ligacoes_icons,array_val,i); e termina a execução do programa.

Quando encontrar um ícone terminador, ícone com 2 ou mais portos de entrada ou ter portos de controlo, começa-se num novo porto de saída do ícone de início de sequência, repetindo-se o algoritmo até não haver portos de saída.

3.18.3 Fluxograma da função Branch_Route

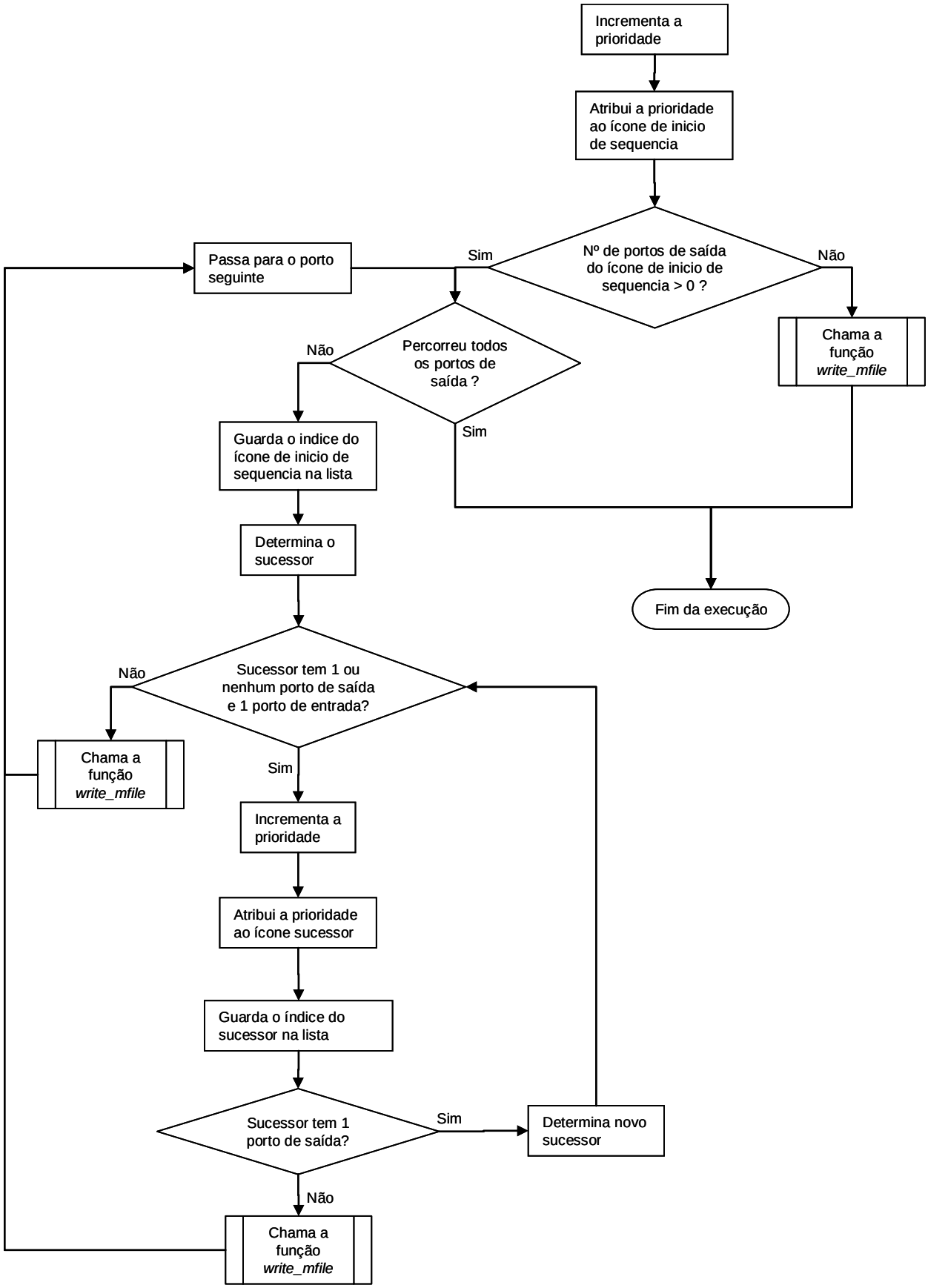


Figura 3.18 Fluxograma da função Branch_Route

3.19 Função *write_mfile*

Esta função tem como objectivo a escrita da m-file com as respectivas funções dos ícones existentes na área de desenho. A escrita desta m-file e feita em simultâneo com as atribuições das prioridades aos ícones existentes na área de desenho, ou seja, no fim de se atribuir as prioridades a cada troço é escrito na função de execução da simulação as respectivas funções dos componentes correspondentes que pertencem a esse troço.

3.19.1 Utilização:

- icons_schematic=write_mfile(icons_schematic, ligacoes_icons,array_val,port_out)

Parâmetros de entrada

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
ligacoes_icons	Estrutura que contém as ligações dos ícones e a quem se encontram ligados.
array_val	Array que indica a sequencia dos ícones no troco
port_out	Índice do porto de saída do ícone de inicio de sequencia.

Parâmetros de saída:

icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.

Ver também: *OSIP_f, run_simulation* e *branch_route*

Help Matlab:

– 'handles'

– 'fopen'

3.19.2 Descrição do algoritmo:

Inicialmente é aberto o ficheiro de execução da simulação em modo de adição de texto e verifica-se se a abertura foi efectuada com êxito.

Em caso afirmativo, é verificado se o ícone actual tem apenas portos de saída. Se sim, é determinado o nome do ícone actual e é escrito no ficheiro o código referente ao componente correspondente e o código de actualização de barra de espera.

No caso do ícone actual ter mais de um porto de entrada e/ou portos de controlo, então é impresso no ficheiro o código referente à leitura da estrutura temporária criada para guardar os sinais. É ainda criado o código referente ao componente correspondente ao ícone actual e actualizada a barra de espera da simulação.

Para o caso dos ícones que não são de início de sequência e têm um ou mais portos de entrada, é escrito no ficheiro o código referente ao componente correspondente e o sinal de saída é passado para a estrutura *sinal* existente para passagem dos sinais entre componentes.

Se o ícone actual não tem portos de saída, o código é escrito sem referência ao sinal de saída.

Para todos os ícones do troço actual, é verificado qual o ícone de destino da ligação do porto de saída actual e feito o *offset* relativo ai tipo de porto (in, Hi ou Lo). De seguida é escrito no ficheiro o código referente ao componente tendo em atenção o porto de destino.

Quando todos os ícones do porto estiverem concluídos, então o programa é terminado, voltando á execução da função *branch_route*.

3.19.2.1 Fluxograma da função write_mfile

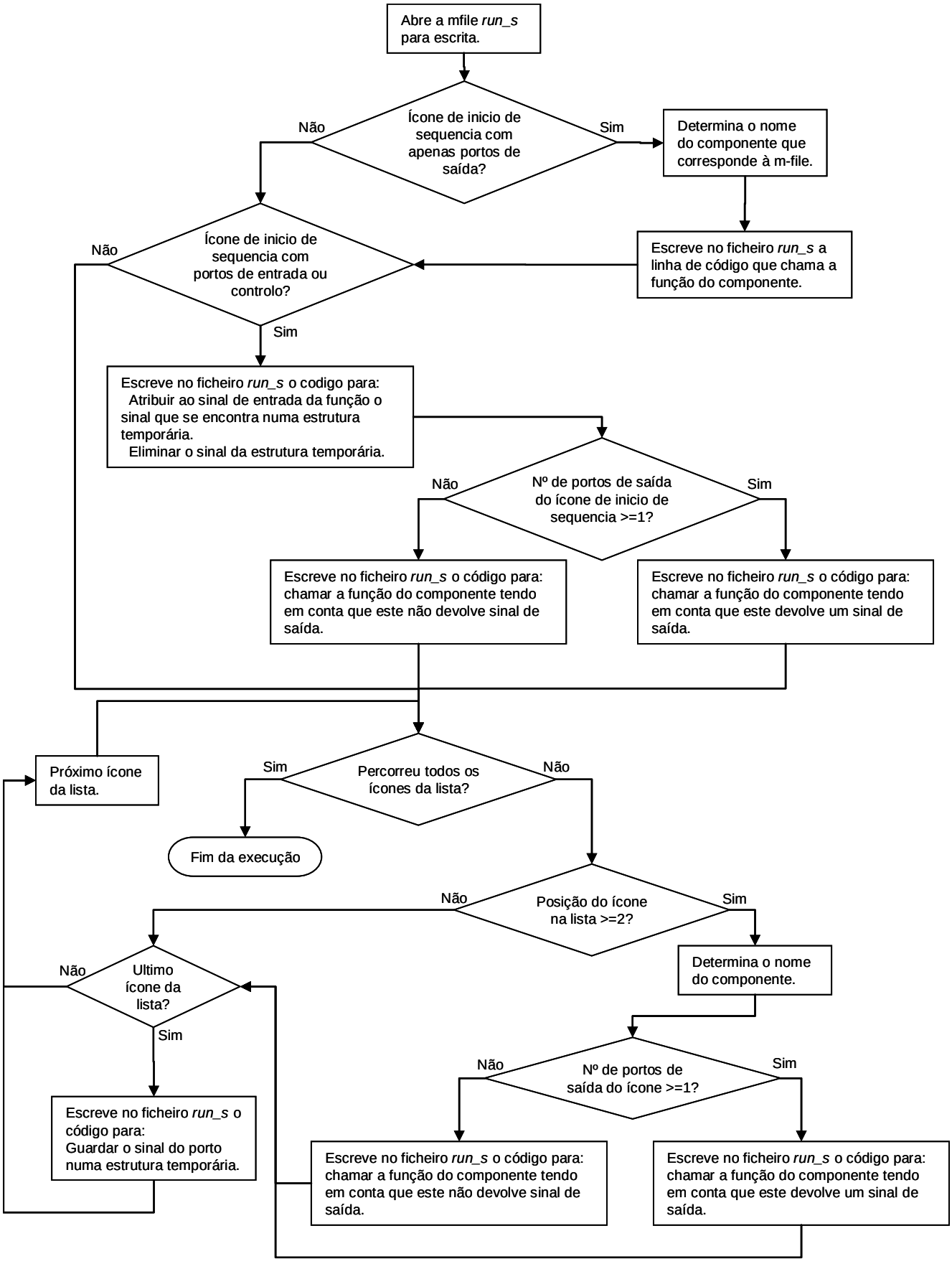


Figura 3.19 Fluxograma da função Branch_Route

3.20 Função *save_function*

Esta função guarda um projecto que se encontra na área de desenho num ficheiro do tipo 'osi'.

Neste ficheiro são guardados os parâmetros que definem um ícone na área de desenho: nome do ícone, o tipo, as coordenadas em x e em y na área de desenho, o número de porto (entrada, controlo e saída), as coordenadas e o tipo de cada porto, qual a tipo do porto que se encontra ligado ao porto de saída, a prioridade do ícone e se os portos de entrada e saída podem ser alterados ou não.

São também guardadas as ligações entre os portos dos diversos ícones existentes na área de desenho e os parâmetros configurados para cada componente.

3.20.1 Utilização:

- save_function(handles)

Parâmetros de entrada:

handles: Estrutura que contém os *handles* dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

nicons

icons_schematic

ligacoes_icons

icons_valid

Ver também: *OSIP_f*, *resizefcn* e *open_function*

Help Matlab:

– 'uiputfile'

– 'struct2cell'

3.20.2 Descrição do algoritmo:

Inicialmente é testado a variável *file_c* de forma a verificar se o utilizador utiliza a operação de guardar o esquemático pela primeira vez. Se se verificar que não é a primeira vez que se guarda o esquemático, este é guardado na pasta e com o mesmo da anterior salvaguarda. O caminho é dado pela variável *path_c* e o nome do ficheiro está contido na variável *file_c.* Se, pelo contrario, verificar-se que é a primeira vez que se guarda o esquemático é criada uma janela onde o utilizador indica o nome e o local do ficheiro onde pretende guardar o projecto através do seguinte comando uiputfile({'Untitled.osi','Save text file (*.osi)'},'Save Work').

De seguida abre-se o ficheiro, para escrita, indicado pelo utilizador imprimindo a hora e a data actual.

Depois guarda-se o comprimento e a altura da janela OSIP.

Posteriormente guardam-se todas as estruturas que contém informação do projecto que se encontra na área de desenho.

Guarda-se ainda a estrutura *nicons* que contém o número de ícones, existentes na área de desenho, e o número de cada tipo de componentes.

Da estrutura *icons_schematic* guarda-se: o nome do ícone, o nome do ícone que este apresenta na área de desenho, o tipo de componente a que o ícone pertence, as coordenadas em *x* e em *y* do ícone na área de desenho, o número de portos (entrada, saída e controlo), se os portos de entrada e saída são alterados ou não, o tipo e as coordenadas em *x* e em *y* de cada porto caso existam.

Da estrutura *ligacoes_icons* guarda-se para todos os ícones existentes na área de desenho as ligações caso existam de todos os portos (entrada, saída e controlo). Como as ligações são constituídas por troços, determina-se as coordenadas em *x* e em *y*, no início e no fim de cada troço da ligação existente na área de desenho. Guarda-se estes valores para quando se fizer a leitura ser possível voltar a redesenhar as ligações.

Por fim são guardados os parâmetros configurados para os ícones que se encontram na área de desenho. Como os parâmetros estão guardados em ficheiros dentro do directório local *\temp*, unicamente se lê todo o conteúdo existente nestes ficheiros e guarda-se o conteúdo no ficheiro *save* indicado pelo utilizador.

3.20.3 Fluxograma da função save_function

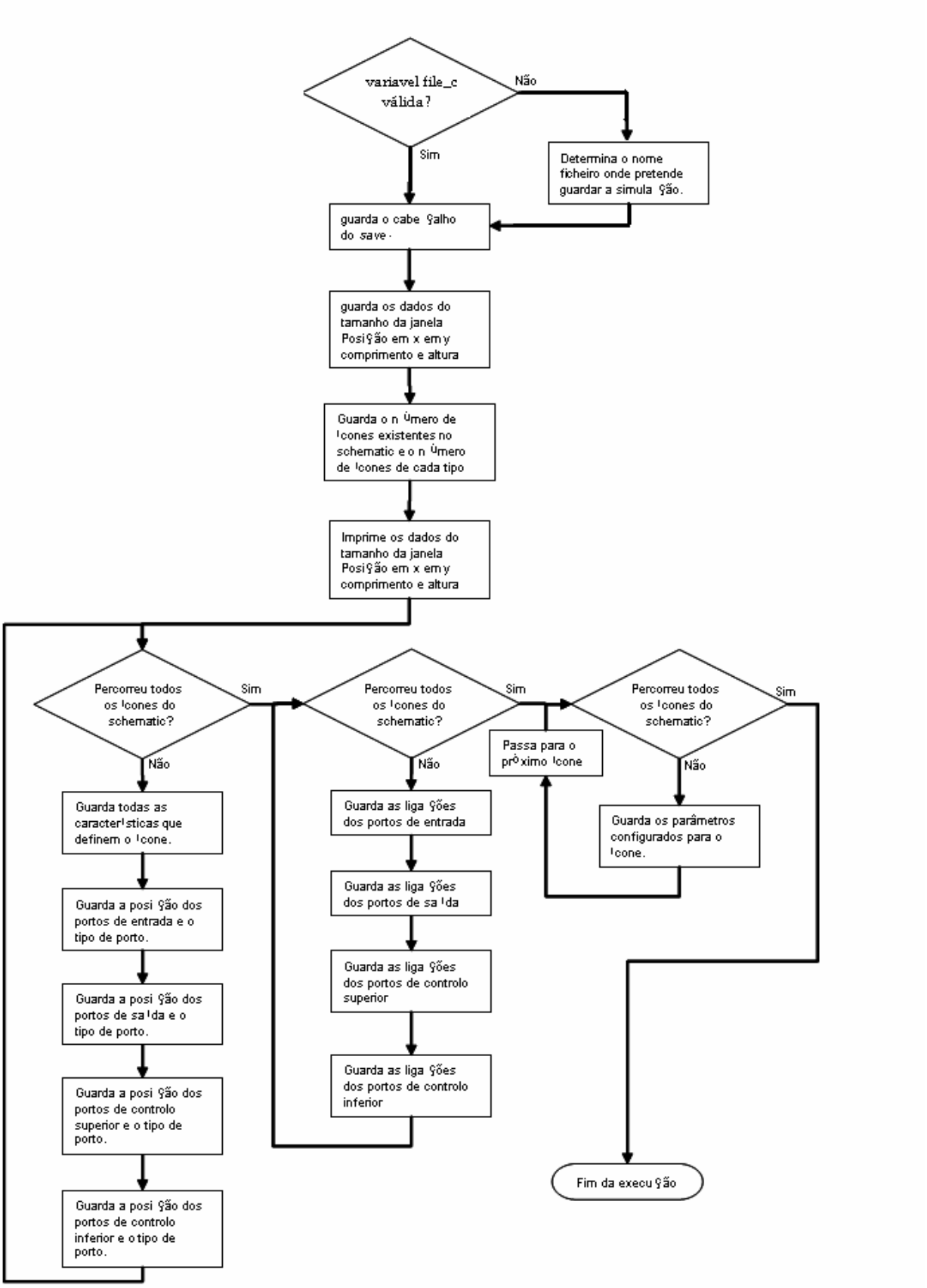


Figura 3.20 Fluxograma da função save_function

3.21 Função *save_as_function*

Esta função executa exactamente a mesma operação que a função anterior (*save_function.m*), mas não averigua se o esquemático foi anteriormente guardado, exigindo sempre ao utilizador que indique o caminho de destino onde se pretende salvar o projecto.

3.21.1 Utilização:

- save_function(handles)

Parâmetros de entrada:

handles: Estrutura que contém os *handles* dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis globais:

nicons

icons_schematic

ligacoes_icons

icons_valid

Ver também: *OSIP_f, resizefcn* e *open_function*

Help Matlab:

– 'uiputfile'

– 'struct2cell'

3.21.2 Descrição do algoritmo:

Inicialmente é criada uma janela onde o utilizador indica o nome e o local do ficheiro onde pretende guardar o projecto através do seguinte comando uiputfile({'Untitled.osi','Save text file (*.osi)'},'Save Work').

De seguida abre-se o ficheiro, para escrita, indicado pelo utilizador imprimindo a hora e a data actual.

Depois guarda-se o comprimento e a altura da janela OSIP.

Posteriormente guardam-se todas as estruturas que contém informação do projecto que se encontra na área de desenho.

Guarda-se ainda a estrutura *nicons* que contém o número de ícones, existentes na área de desenho, e o número de cada tipo de componentes.

Da estrutura *icons_schematic* guarda-se: o nome do ícone, o nome do ícone que este apresenta na área de desenho, o tipo de componente a que o ícone pertence, as coordenadas em *x* e em *y* do ícone na área de desenho, o número de portos (entrada, saída e controlo), se os portos de entrada e saída são alterados ou não, o tipo e as coordenadas em *x* e em *y* de cada porto caso existam.

Da estrutura *ligacoes_icons* guarda-se para todos os ícones existentes na área de desenho as ligações caso existam de todos os portos (entrada, saída e controlo). Como as ligações são constituídas por troços, determina-se as coordenadas em *x* e em *y*, no início e no fim de cada troço da ligação existente na área de desenho. Guarda-se estes valores para quando se fizer a leitura ser possível voltar a redesenhar as ligações.

Por fim são guardados os parâmetros configurados para os ícones que se encontram na área de desenho. Como os parâmetros estão guardados em ficheiros dentro do directório local *temp*, unicamente se lê todo o conteúdo existente nestes ficheiros e guarda-se o conteúdo no ficheiro *save* indicado pelo utilizador.

3.21.3 Fluxograma da função save_as_function

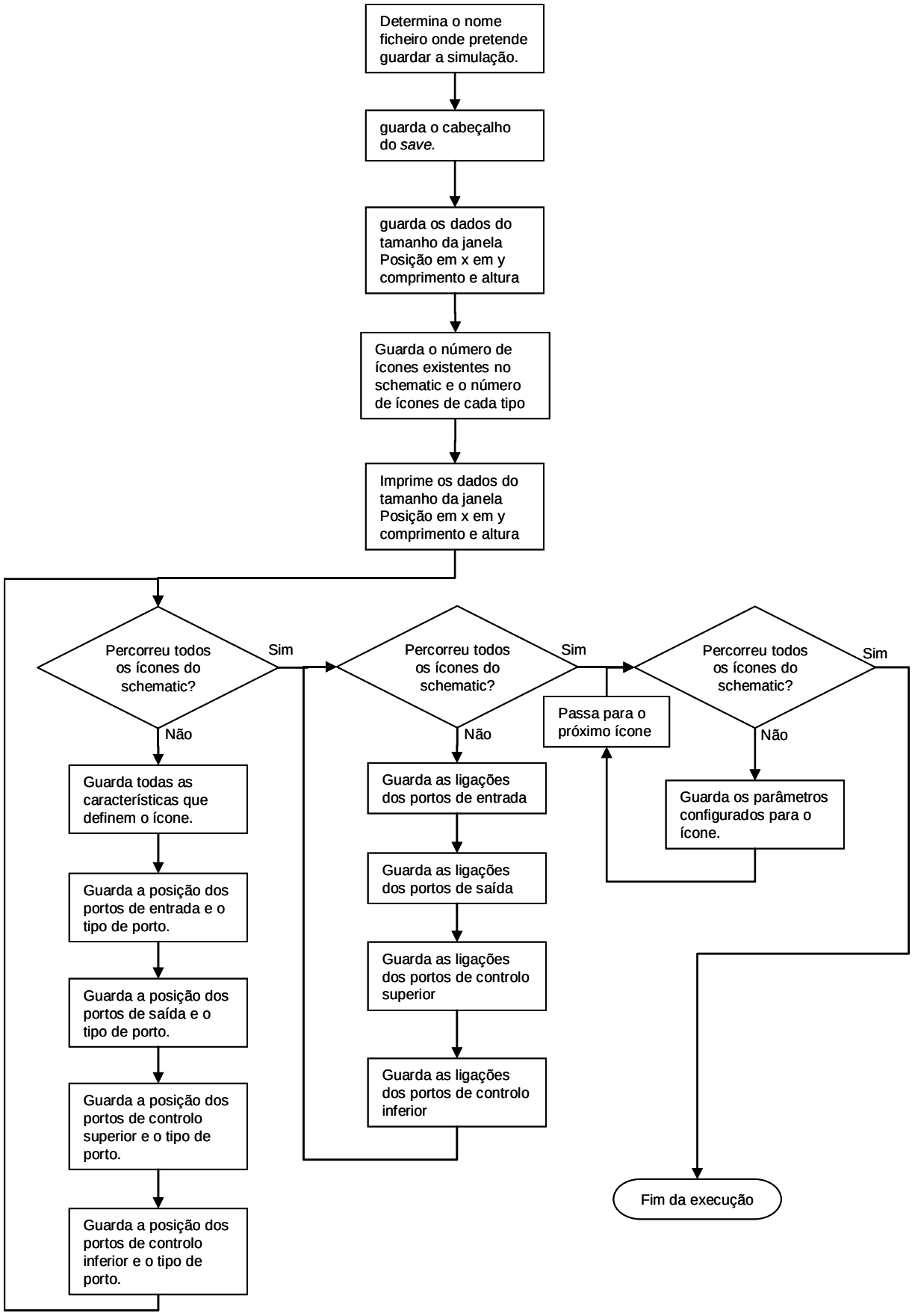


Figura 3.21 Fluxograma da função save_as_function

3.22 Função *open_function*

Esta função faz a abertura de uma simulação que foi previamente guardada. As simulações são guardadas em ficheiros com a extensão "osi". Estes ficheiros contêm toda a informação a cerca dos componentes, ligações entre os vários componentes e o valor dos parâmetros configurados para cada componente presentes na área de desenho.

3.22.1 Utilização:

- open_function(handles)

Parâmetros de entrada

handles Estrutura que contém os *handles* dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis Globais:

nicons

icons_schematic

ligacoes_icons

icons_valid

Ver também: *OSIP*, *resizefcn* e *save_function*

Help Matlab:

- 'handles'
- 'uigetfile'
- 'num2cell'
- 'cell2struct'

3.22.2 Descrição do algoritmo:

Inicialmente são declaradas e inicializadas as variáveis globais e locais da função. De seguida é criada a janela que permite ler o nome do ficheiro de save do projecto a ser lido – [file, path] = uigetfile({'*.osi','Saved files (*.osi)'}, 'Open');.

De seguida é verificado se a o nome do ficheiro é válida. Se sim, o ficheiro é aberto para leitura e descarta as linhas iniciais (são linhas de referência que contém informação adicional sobre o projecto).

De seguida é feita a leitura do tamanho da janela e feito o redimensionamento da janela através da função: resizefcn(handles,flag_open,window);

Seguidamente é feita a leitura e a actualização das estruturas *nicons*, *icons_schematic e ligacoes_icons.*

São ainda desenhadas as imagens dos componentes e os respectivos portos na área de desenho nas posições guardadas e com as respectivas ligações.

Por fim são lidos os valores dos parâmetros (podem ser os valores por defeito ou alterados) de cada um dos componentes desenhados na área de desenho.

O ficheiro á fechado e termina a execução do programa.

3.22.3 Fluxograma da função open_function

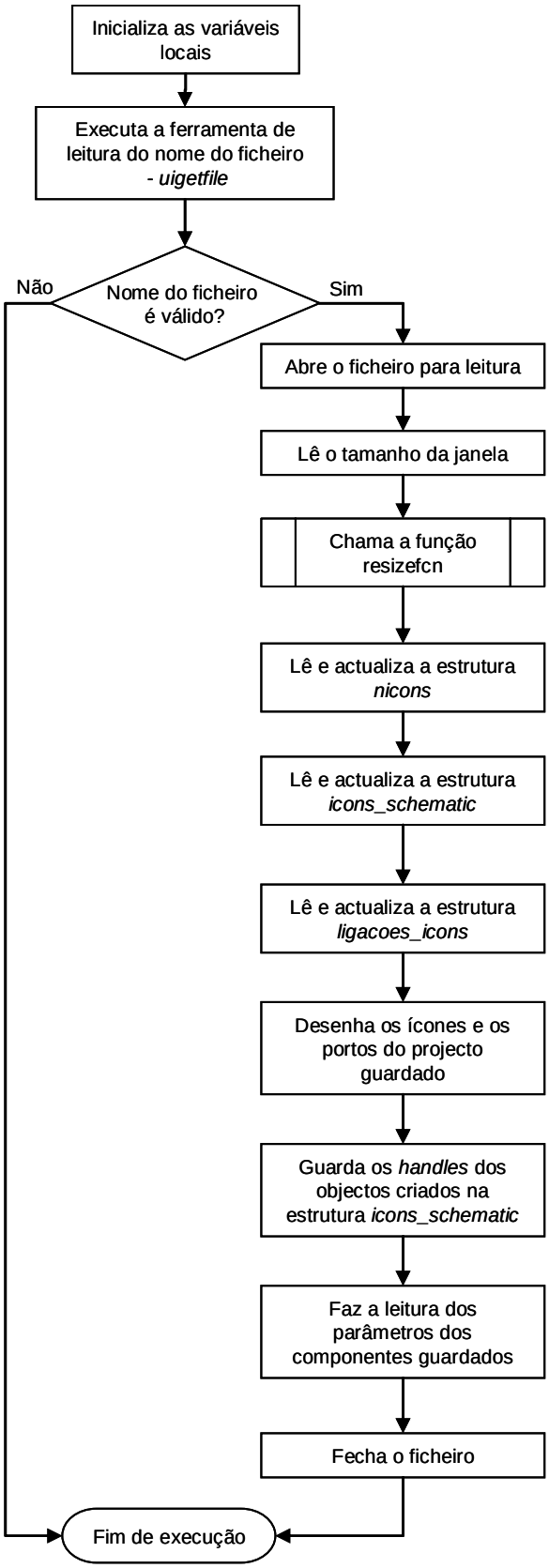


Figura 3.21 Fluxograma da função save_function

3.23 Função *resizefcn*

Esta função faz o ajuste da área de desenho quando se aumenta o tamanho da janela OSIP.

O tamanho mínimo da janela em x é *width* = 970 pixels e em y height = 670 pixels.

3.23.1 Utilização:

- resizefcn(handles,flag_open,length_window)

Parâmetros de entrada:

handles	Estrutura que contém os *handles* dos objectos da janela OSIP.
flag_open	Flag que indica quando a 1, que se abriu o projecto de uma simulação que foi previamente guardado.
length_window	Array que contém o tamanho da janela quando se faz a leitura de um ficheiro de um projecto guardado. Em todos os outros casos deve ser um vector nulo.

Parâmetros de saída:

Não existem.

Variáveis globais:

Não existem.

Ver também: *osip_f*

Help Matlab:

– 'xlim'

– 'Position'

3.23.2 Descrição do algoritmo:

Inicialmente determina-se: o tamanho e a posição da janela OSIP, o tamanho e a posição da área de desenho e a posição dos *slider*.

A seguir verifica se pretende redimensionar a janela quando se faz a abertura de um projecto guardado, para o tamanho da janela igual aquando se guardou o projecto. Em caso afirmativo redimensiona o tamanho da janela para os valores indicados pelo parâmetro *length_window* através do seguinte comando: set(handles.osip_f,'Position',osip_f_position). Caso contrário verifica se o *resize* da janela é inferior aos limites mínimos e em caso afirmativo redimensiona a janela para os limites mínimos (970 X 670 pixels).

De seguida verifica-se se o comprimento da janela OSIP é maior que o limite mínimo, em caso afirmativo altera-se a área de desenho a posição e o tamanho do *slider* horizontal para um valor dependente do tamanho da janela OSIP. Caso contrário altera-se a área de desenho, a posição e o tamanho do *slider* par os respectivos valores mínimos (comprimento mínimo da área de desenho 800 pixels e o comprimento mínimo do *slider* horizontal 802 pixels).

Por fim verifica-se se a altura da janela OSIP é maior que o limite mínimo e aplica-se o mesmo procedimento descrito anteriormente para o caso do comprimento.

3.23.3 Fluxograma da função resizefcn

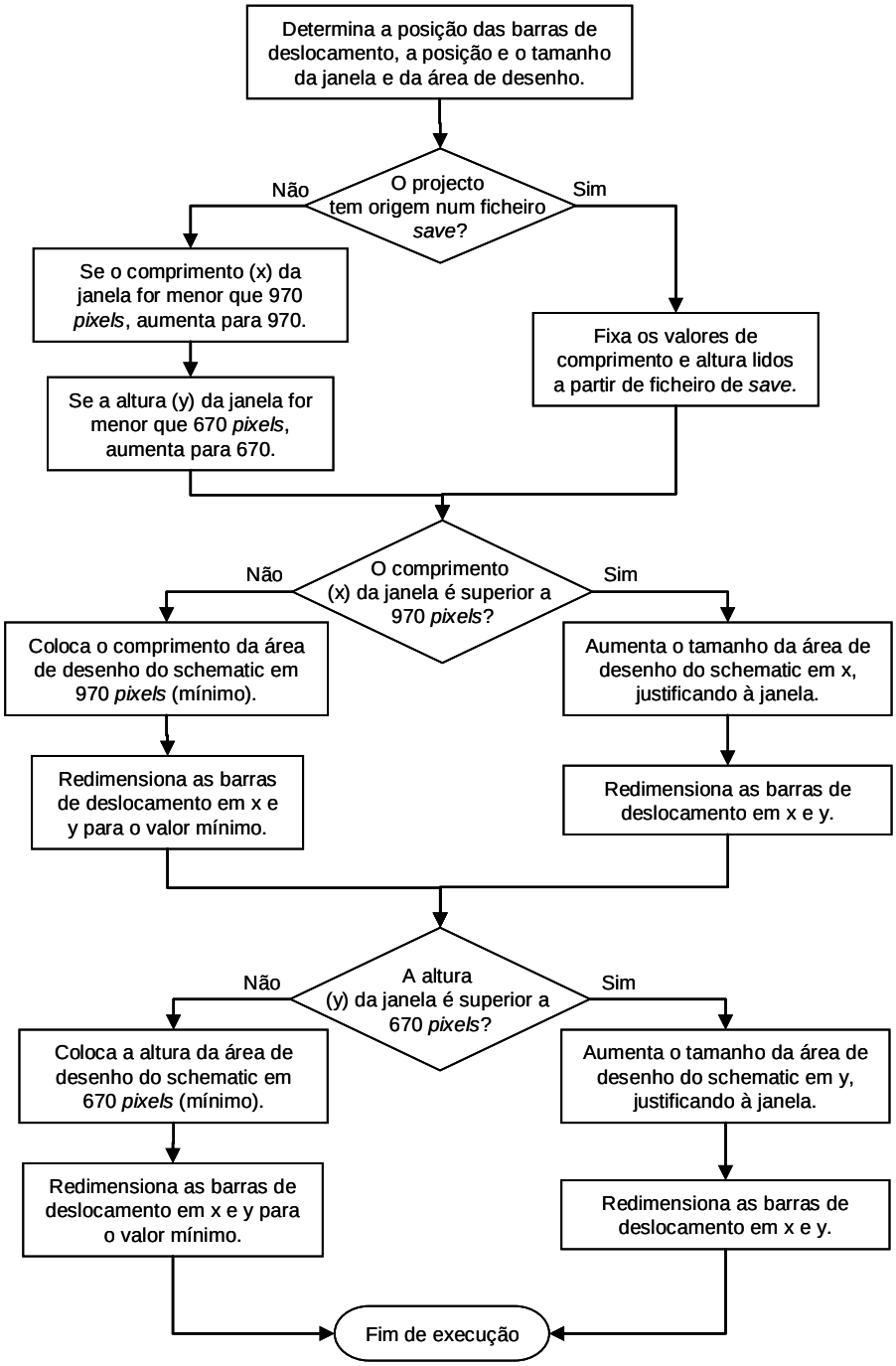


Figura 3.22 Fluxograma da função resizefcn

3.24 Função *undo*

Esta função permite retroceder na elaboração de um esquemático. Para isso é realizada uma operação de *save* do esquemático sempre que é efectuada uma das seguintes operações:

- Deslocação de um componente no esquemático;
- Alteração de parâmetros de um componente;
- Ligação entre dois componentes;
- Apagar ligações;

Ao efectuar *undo* apenas se vai abrir o esquemático anteriormente guardado. Se forem realizadas várias operações de *undo* sequencialmente são abertos os esquemáticos pela ordem de salvaguarda. Apenas é possível, nesta fase, realizar um processo de retrocesso (*undo*) 4 vezes consecutivas.

3.24.1 Utilização:

- undo_function(handles)

Parâmetros de entrada

handles Estrutura que contém os *handles* dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis Globais:

Und

Ver também: *OSIP*, *mouseclickdown*, *keypressfunc*, *new_project*, *undo_open1_function e undo_save_function*

Help Matlab:

– 'handles'

3.24.2 Descrição do algoritmo:

A função *undo* é construída tendo por base as funções de *save* e *open* do simulador. Sempre que se toma as acções mais importantes dentro no esquemático do

simulador (ver utilização) é chamada a função *undo_function.m* e é tomada a decisão sobre qual operação se deve realizar.

É executada a operação de *Save* sempre que, pela acção do rato ou pela acção do teclado, se altera o esquemático. Nas funções *mouseclickdown.m* e *keypressfunc.m* vai ser activada a flag *clic_flag* e chamada a função *undo_function.m* que ao testar a referida flag executa a função *undo_save_function.m*. Esta função irá salvaguardar o esquemático na pasta *Temp*, na posição 0 destinada ao *undo* (*..\temp\undosave_1*). Ao continuar a invocar as funções *mouseclickdown.m* e *keypressfunc.m* (pela acção do rato / teclado), o esquemático será guardado posições seguintes do *undo* (1, 2e 3), sendo que na quinta salvaguarda esta será realizada de novo na posição 1. Para uma melhor percepção aconselha-se a visualização da parte esquerda da figura 3.23.

A operação de *open* é utilizada sempre que se pretenda retroceder no esquemático, e portanto abrir os esquemáticos anteriormente guardados pela ordem sequencial. Nesta situação é chamada a função *undo_function.m* com a flag *undo_flag* activada, o que invoca a função *undo_open1_function.m*, que irá realizar a abertura do esquemático anteriormente guardado. Caso se continue a retroceder é aberto o esquemático anterior até ao máximo de 4 esquemáticos.

3.24.3 Fluxograma da função undo

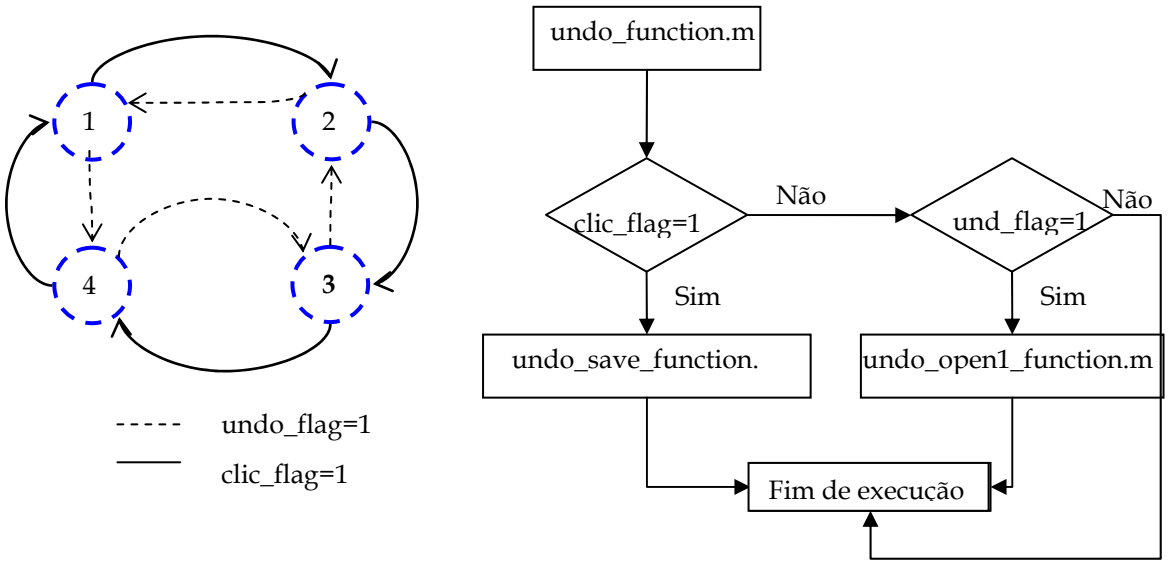


Figura 3.23 Fluxograma da função undo

3.25 Pasta *Hierarchy*

Esta pasta contém as diversas funções que realizam o agrupamento de componentes no esquemático. As diversas funções são chamadas de forma sequencial conforme se apresenta no fluxograma geral, de forma a criar um novo componente no simulador, com as funções executáveis dos componentes que o compõem.

3.25.1 Utilização:

- creat_component_hierarchy(handles) [-> 1ªfunção da sequência]

Parâmetros de entrada

handles Estrutura que contém os *handles* dos objectos da janela OSIP.

Parâmetros de saída:

Não existem.

Variáveis Globais:

IconsSelected

icons_valid

icons_schematic

ligacoes_icons

NUM

matriz_in

matriz_out

Ver também: *OSIP*, *creat_component_hierarchy2*, *creat_comp_in*, *creat_comp_out*, *hierarquia_open3*, *hierarquia_save3*.

Help Matlab:

– 'handles'

3.25.2 Descrição do algoritmo:

A função de agrupar componentes é composta de diversas funções auxiliares que realizam a inserção de novos módulos no simulador, através da junção das diversas *m-files* referentes aos componentes agrupados. As diversas funções que realizam estas operações estão contidas na pasta *Hierarchy* dentro do directório *kernel*.

Esta funcionalidade de agrupar componentes apenas é activada quando no esquemático se encontram dois ou mais componentes seleccionados. A operação de activação é realizada na função *mouseclickup.m* e desactivada na função *mouseclickdown.m*. Convém ainda advertir o utilizador que os portos dos componentes destinados a portos de entrada /saída do novo componente (agrupado) devem estar sem ligações a outros componentes fora da selecção activas.

Após selecção dos componentes, o primeiro passo é o de configuração dos parâmetros do novo componente, com a inserção do nome, pasta de destino dentro do directório modules, icon que é atribuído e possíveis comentários. Esta operação é realizada pela função *hierarchy1.m* que é chamada pela função principal *creat_component_hierarchy.m*.

O passo seguinte é a configuração dos portos de entrada. É criada uma janela de configuração dos portos de entrada dos componentes seleccionados, os quais não possuem ligações. São mostrados, no máximo, três portos de entrada de três componentes seleccionados com portos de entrada nas condições referidas. Após a visualização é dado ao utilizador a possibilidade de optar, através de *check's box*, pelos portos pretendidos para portos de entrada do novo componente. Este passo é realizado com a chamada da função *creat_comp_in.m* contida dentro da pasta *Hierarchy*.

De seguida configura-se os portos de saída da mesma forma que os de entrada. Igualmente são mostrados os portos de saída dos componentes seleccionados num máximo de três portos de saída referentes a três componentes diferentes. Para a selecção dos portos de saída é chamada a função *creat_comp_out.m*.

Após a selecção dos portos de saída são criadas todas as *m-files* e ficheiros necessários ao funcionamento do novo componente:

- Criação da *m-file* principal com a chamada de todas as funções dos componentes que compõem o grupo representado pelo novo componente;
- Criação da *m-file* *_f.m responsável pela criação de uma janela de interface com o utilizador onde se mostra e se tem possibilidade de aceder à janela de configuração de parâmetros dos vários componentes que formam o componente. Através desta é possível editar e alterar os diversos parâmetros destes componentes.
- Criação do ficheiro *.txt para o directório *components* que possui todas as informações necessárias sobre os portos do novo componente.
- Criação do ficheiro *.txt para o directório *Default Definitions* que possui um campo com a *string* 'hierarchy' que identifica o componente como um grupo de componentes; possui campos com o nome de cada componente que constitui o grupo e ainda um último campo que indica o número de componentes do grupo. Estes campos são úteis para a posterior salvaguarda de esquemáticos que contenham componentes.
- Ainda é crida a possibilidade do utilizador visualizar o interior do componente global, com a salvaguarda da parte seleccionada do esquemático em *..\temp\hierarquia\teste.mat*. A partir deste ficheiro é possível ao utilizador verificar todo o conjunto de componentes agrupado através da selecção do *push button 'components'* a partir da janela de parâmetros.

Todas estas funcionalidades e implementações descritas neste último passo encontram-se realizadas na função *creat_component_hierarchy2.m*.

3.25.3 Fluxograma da funcionalidade agrupar componentes

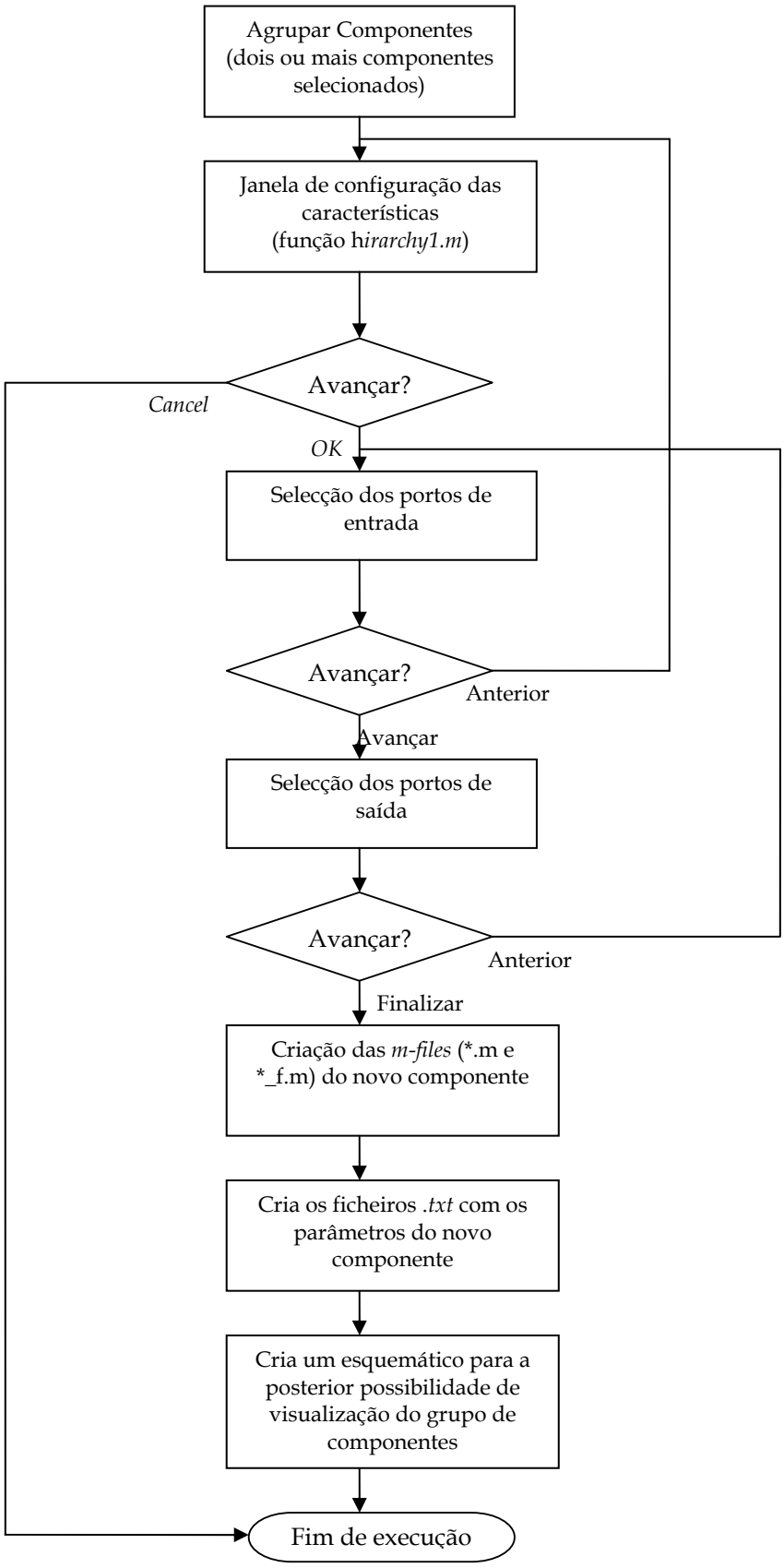


Figura 3.24 Fluxograma da criação de grupos de componentes

Capitulo 4

Funções no directório Import

As funções presentes neste directório tem como finalidade efectuar a importação ou exportação de componentes do simulador.

Existem dois tipos de importação. Pode-se importar um componente em que o modelo encontra-se descrito por uma *m-file* ou importar um componente que já foi previamente adicionado noutro computador ao simulador.

4.1 Função *import_component*

Esta função foi construída através da ferramenta de construção gráfica do Matlab GUIDE. Tem como objectivo cria a janela de importação dos componentes para o simulador, a importação está dividida em três passo: no primeiro passo são descritas o local onde se encontra a *m-file* do modelo do componente, qual o tipo do componente, a imagem para o componente, o nome do sinal de entrada e o nome do sinal de saída da *m-file*. No segundo passo são configurados os portos do componente, número de portos de entrada, saída e controlo, e o tipo de porto optical, electrical, logical e value. No terceiro e último passo são configurados os parâmetros do componente.

Parâmetros de entrada:

Não existem.

Parâmetros de saída:

Não existem.

Variáveis globais:

Não existem.

4.1.1 Descrição do algoritmo:

A função *import_component*, como foi construída através da ferramenta GUIDE do Matlab, encontra-se subdividida em várias funções locais, que são executadas dependendo das acções tomadas.

A função local *varargout = import_component(varargin)* faz a inicialização do código e cria a janela de interface com o utilizador.

A função *import_component_OpeningFcn(hObject, eventdata, handles, varargin)* faz a abertura da função *import_component*, nesta função é inicializada a estrutura onde vão ser guardados os dados introduzidos pelo utilizador handles.var_step1=[]; handles.var_step2=[]; handles.var_step3_2=[]; e a flag de controlo de erros handles.flag_error=0,

para que as alterações na estrutura *handles* também sejam vistas nas outras funções tem que se fazer o *update* da estrutura através do seguinte comando guidata(hObject, handles).

A função *varargout = import_component_OutputFcn (hObject, eventdata, handles)* faz a saída de dados da função, como a função *import_component* não têm parâmetros de saída esta função devolve um valor vazio varargout{1} = {}.

Função *pushbutton_step1_Callback (hObject, eventdata, handles)*, esta função abre a janela de configuração dos parâmetros do *Step1*, da janela de importação do componente quando se pressiona o botão *OK* correspondente, e guarda os parâmetros configurados na variável *handles.var_step1*. Os parâmetros configurados são: o nome do componente, o local onde se encontra a *m-file* do modelo do componente, o tipo de componente, o nome do sinal de entrada e saída da *m-file*. De seguida activa o botão de *OK* para configurar os parâmetros do *Step2* através do seguinte comando: set(handles.pushbutton_step2,'Visible','on').

Função *pushbutton_step2_Callback (hObject, eventdata, handles)*, esta função tem a mesma funcionalidade que a função descrita para o *Step1*, com a diferença que os parâmetros configurados nesta janela são: o número de portos de entrada, saída e controlo, se o número de portos de entrada e saída é fixo ou podem ser alterados e o tipo de cada porto (optical, electrical, logical e value). Também é activado o botão de *OK* para configurar os parâmetros do *Step3*.

Função *pushbutton_step3_Callback (hObject, eventdata, handles)* tal como a função descrita para o *Step1* e *Step2* abre a janela de configuração dos parâmetros do *Step3*. A finalidade desta janela é configurar os parâmetros internos do componente, tara tal são descritos: o nome do parâmetro, a dimensão (unidimensional ou multidimensional), as unidades, campo que não se preenche caso não se aplica, a natureza do parâmetro (Integer double ou string), o estilo de visualização, só existe caso o parâmetro seja multidimensional, o valor ou valores por defeito e o comentário acerca do parâmetro.

A função *pushbutton_OK_Callback(hObject, eventdata, handles)* verifica se os parâmetros introduzidos nos três passos são correctos e cria o componente no simulador através das funções: *creat_files_component(var_step1,var_step2,var_step3)* cria os ficheiros com os parâmetros do componente e os seus valores por defeito, *creat_component_schematic(var_step1,var_step3)* adiciona o componente ao simulador e a

função *creat_component_parameters(var_step1,var_step3)* cria a janela de configuração dos parâmetros do componente. Por fim fecha a janela de importação do componente através do seguinte comando: close (handles.import_component).

Por ultimo a função pushbutton_Cancel_Callback(hObject, eventdata, handles) interrompe a importação do componente e fecha a janela.

4.1.2 Fluxograma da função import_component

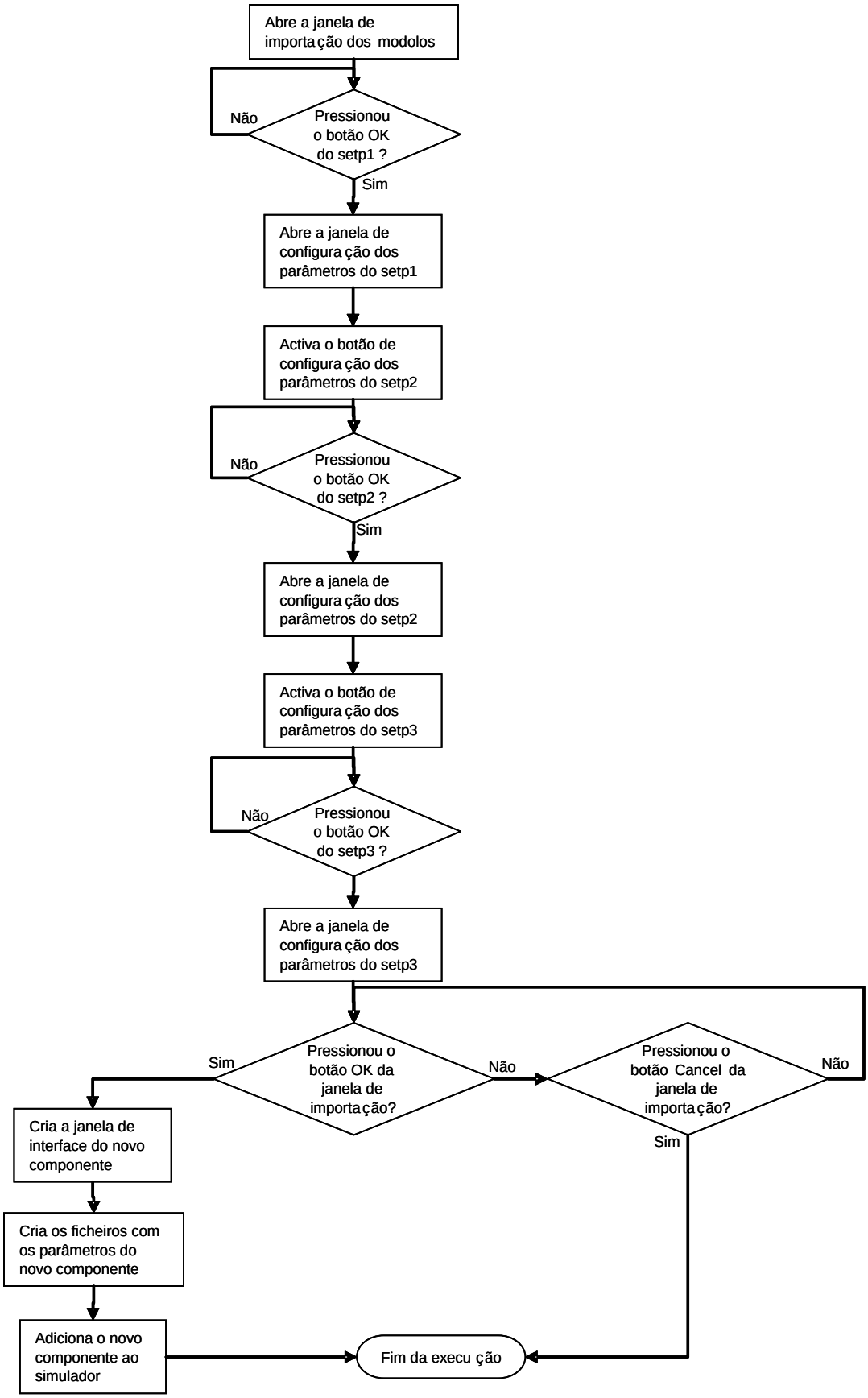


Figura 4.1 Fluxograma da função import_component

4.2 Função *export_file_component*

Esta função foi construída com a ferramenta gráfica do Matlab GUIDE. E tem como objectivo fazer a exportação de componentes existentes no simulador para poderem ser usados no simulador noutro computador.

4.2.1 Utilização:

- export_file_component

Parâmetros de entrada:

Não existem.

Parâmetros de saída:

Não existem.

Variáveis globais:

Não existem.

Ver também: *import_file_component* e *import_component*

Help Matlab:

– 'GUIDE'

– 'handles'

4.2.2 Descrição do algoritmo

A função *export_file_component*, como foi construída através da ferramenta GUIDE do Matlab, encontra-se subdividida em várias funções locais, que são executadas dependendo das acções tomadas.

A função local *varargout = export_file_component(varargin)* faz a inicialização do código e cria a janela de interface com o utilizador.

A função *export_file_component_OpeningFcn(hObject, eventdata, handles, varargin)* faz a abertura da função *export_file_component*.

A função *export_file_component* cria uma janela onde se pede ao utilizador para indicar qual o componente que se pretende exportar e o local onde vai ser guardado o ficheiro do tipo 'oic'. Este é um ficheiro comprimido que contém as m-file do modelo do componente, a m-file que cria a janela de configuração dos parâmetros, a imagem do componente e os ficheiros de texto com os valores por defeito dos parâmetros e definições dos portos.

4.3 Função *import_file_component*

Esta função foi construída com a ferramenta gráfica do Matlab GUIDE. E tem como objectivo fazer a importação de componentes que já foram previamente adicionados noutro computador ao simulador.

4.3.1 Utilização:

- import_file_component

Parâmetros de entrada:

Não existem.

Parâmetros de saída:

Não existem.

Variáveis globais:

Não existem.

Ver também: *export_file_component* e *import_component*

Help Matlab:

– 'GUIDE'

– 'handles'

4.3.2 Descrição do algoritmo

A função *import_file_component*, como foi construída através da ferramenta GUIDE do Matlab, encontra-se subdividida em várias funções locais, que são executadas dependendo das acções tomadas.

A função local *varargout = import_file_component(varargin)* faz a inicialização do código e cria a janela de interface com o utilizador.

A função *import_file_component_OpeningFcn(hObject, eventdata, handles, varargin)* faz a abertura da função *import_file_component*.

A função *import_file_component* cria uma janela onde se pede ao utilizador para indicar qual o ficheiro do tipo '*oic*', que contém o componente que se pretende importar, e o local onde este vai ser guardado no directório dos componentes no simulador.

O ficheiro lido é um ficheiro comprimido que contém as m-file do modelo do componente, a m-file que cria a janela de configuração dos parâmetros, a imagem do componente e os ficheiros de texto com os valores por defeito dos parâmetros e definições dos portos.

Cada um destes ficheiros é colocado nos respectivos directórios.

Capitulo 5

Directório Suporte

No directório suporte estão guardados os ficheiros que servem de suporte aos componentes existentes no simulador. A maior parte destas funções foi desenvolvida e explicado na versão anterior do simulador OSIP, desenvolvida pelos colegas Ana Ferreira e Tiago Silveira no seu projecto de final de curso [1].

Neste ponto são explicadas as funções necessárias ao funcionamento da interface gráfico do simulador:

5.1 Função *Port_definitions*

Esta função é responsável pela criação da interface de configuração dos portos do componente que faz a sua chamada. É ainda responsável por ler o tipo e o número de

portos do componente. Neste ponto os portos configuráveis são os de entrada e de saída, sendo os de controlo (caso existam) fixo.

No directório existe um conjunto de funções com nome *Port_definitions* e terminadas por um caracter. Este caracter corresponde ao componente alvo a que se destina esta função, ou seja, a função *Port_definitions_o.m* é chamada apenas por componentes que tenha apenas entradas ópticas e saídas ópticas. Assim a correspondência dos caracteres terminais é a seguinte:

…_c.m Portos do componente **c**ombinados entre os diversos tipos. Os portos de entrada e saída podem ser do mesmo tipo ou de tipos diferentes.

…_o.m Portos do componente **ó**pticos. Os portos de entrada e saída são do tipo óptico.

…_e.m Portos do componente **e**léctricos. Os portos de entrada e saída são do tipo eléctrico.

…_l.m Portos do componente **l**ógicos. Os portos de entrada e saída apenas aceitam sequências binárias.

…_v.m Portos do componente **v**alor. Os portos de entrada e saída apenas aceitam strings ou valores.

5.1.1 Descrição do algoritmo

Ao se chamar a função *Port_definitions* é passado o nome do componente que a chama. De seguida são inicializadas as estruturas globais e locais da função.

O programa começa por ler os parâmetros actuais dos portos desse componente e coloca-os na janela de interface (número de portos e tipo).

De seguida e de cada vez que o utilizador alterar um valor (ou string) na janela de interface, é verificado esse valor e validado. Caso contrário é enviada uma mensagem de erro a indicar que o valor/string introduzido é inválido.

Por fim se o utilizador pressionar o botão 'OK', são novamente verificados os campos da interface gráfica, guarda-os no ficheiro temporário existente no directório t*emp* e actualiza todas as estruturas dependentes do número e tipo de portos

(icons_schematic e ligacoes_icons). Por fim faz o redesenho dos portos do componente caso o numero de portos tenha sido alterado.

Se o utilizador pressionar no botão 'cancel' é fachada a janela sem efectuar nenhuma alteração.

5.2 Função *ValueDisplayListbox*

Esta função faz a visualização de um sinal do tipo 'value' numa janela do Matlab.

5.2.1 Utilização:

- ValueDisplayListbox(sinal_in,FigName)

Parâmetros de entrada

sinal_in	O sinal de entrada é uma estrutura, ou um array de estruturas, que tem três campos.
sinal_in(i).x	Array de células que contém os nomes das variáveis do porto 'i'.
sinal_in(i).y	Array de células que contem o valor das variáveis descritas em x.
sinal_in(i).type	Tipo do sinal de entrada.
sinal_in(i).noise_DSP:	Componente DEP do ruído à entrada
sinal_in(i).bin:	Sequência binária inicial
sinal_in(i).bit_rate:	Ritmo de transmissão
sinal_in(i).fsignal:	Frequência de referência do sinal
FigName	Nome da janela.

Ver também: *osip_f, ValueDisplay*

Help Matlab:

– 'listbox'
– 'uicontrol'

5.2.2 Descrição do algoritmo

Inicialmente são inicializadas as variáveis locais. De seguida é feito a criação da janela com uma listbox cujos parâmetros são fixos e definidos por: handle_fig=figure('position',[200 200 500 200], 'Resize','off','MenuBar','none','Tag',NameTag,'name', FigName,'NumberTitle','off');.

De seguida é definido o tipo de letra e é colocado a *listbox* vazia (é necessário iniciar a *listbox*). De seguida para todos os sinais de entrada do componente *ValueDisplay* é lido o valor e o nome do sinal em questão.

Por fim os nomes dos sinais e os seus valores são adicionados à listbox e termina o programa.

5.3 Função *Help_win*

Esta função executa a janela de visualização da descrição de um dado componente. É chamada na função de criação da janela de configuração dos parâmetros do componente (*_f.m) sempre que o utilizador faça uso do botão *help*. Para além da descrição do componente, é ainda dada a possibilidade de aceder directamente à página referente ao componente, do *help* do simulador.

5.3.1 Utilização:

- help_wind(handles.index,handles.schematic,handles.janela_schematic)

Ver também: *osip_f*

5.3.2 Descrição do algoritmo

Inicialmente são inicializadas as variáveis locais. De seguida é criada a janela onde é mostrada uma frase contendo uma descrição simples da funcionalidade do

componente. Esta frase é definida em cada componente no ficheiro *_f.m, numa variável de nome *ajuda*.

Nesta função são ainda criados dois *pushbuttons*. O *OK* apenas fecha a janela de visualização e o *Manual* abre a janela do manual referente ao componente em questão. Esta janela do manual encontra-se inserida na pasta Help.

Capitulo 6

Directório Modules

Neste directório contém 14 sub-directórios onde se encontram os ficheiros associados aos componentes existentes no simulador.

Cada sub-directório contém um conjunto de m-files. Estas m-files contêm os programas dos modelos matemáticos dos componentes. Assim, para cada componente existe uma m-file que obedece às regras de entradas e saídas pelas estruturas *sinal_in* e *sinal_out*, sendo que o cabeçalho das funções é do tipo:

function sinal_out= <*nome do componente*>(NUM,sinal_in,icons_schematic,name,port_index, struct_index)

Caso os componentes não tenham parâmetros de saída o cabeçalho das funções passa a ser:

function<*nome do componente*>(NUM,sinal_in,icons_schematic,name,port_index,struct_index)

Onde temos os seguintes parâmetros:

Parâmetros de entrada

NUM	Estrutura que contem os parâmetros numéricos da simulação.
sinal_in	O sinal de entrada é uma estrutura, ou um array de estruturas, que tem três campos.
sinal_in(i).x	Sinal em x.
sinal_in(i).y	Sinal em y.
sinal_in(i).type	Tipo do sinal de entrada.
sinal_in(i).noise_DSP:	Componente DEP do ruído à entrada
sinal_in(i).bin:	Sequência binária inicial
sinal_in(i).bit_rate:	Ritmo de transmissão
sinal_in(i).fsignal:	Frequência de referência do sinal
icons_schematic	Estrutura que contém informação dos ícones presentes na área de desenho.
name	Nome do ícone no schematic que se pretende executar.
port_index	(não se aplica nesta versão; aplicava-se na versão 2.1).
struct_index	Posição do ícone na estrutura icons_schematic.
FigName	Nome da janela.

Parâmetros de saída:

sinal_out	O sinal de saída é uma estrutura, ou um array de estruturas, que tem três campos.
sinal_out(i).x	Sinal em x.
sinal_out(i).y	Sinal em y.
sinal_out(i).type	Tipo do sinal de entrada.
sinal_out.noise_DSP:	Componente DEP do ruído à saída
sinal_out.bin:	Sequência binária inicial

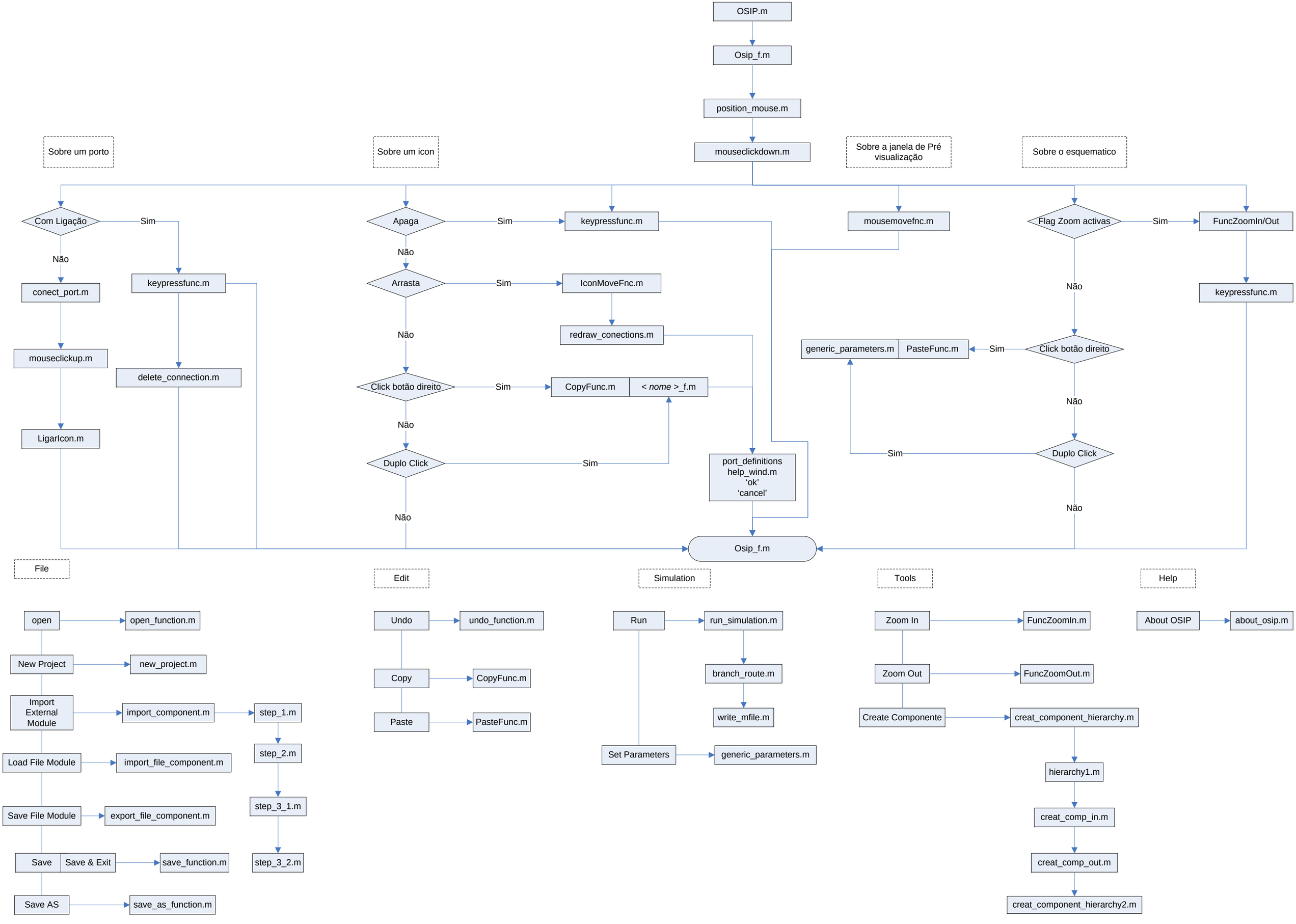
sinal_out.bit_rate:	Ritmo de transmissão
sinal_out.fsignal:	Frequência de referência do sinal

Existem ainda os ficheiros do componente responsáveis por criar a janela de interface de configuração dos parâmetros do componente. Esses ficheiros são terminados em "*_f.m" e "*_f.fig" e são responsáveis por criar a janela de interface, validar os valores introduzidos nos campos existentes e guardar esses valores no ficheiro temporário criado no directório *temp*, enquanto o ícone do componente não for apagado da área de desenho do simulador.

Capitulo 7

Anexo

Apresenta-se na página seguinte um fluxograma geral de todo o simulador OSIP. Nele estão representadas todas as funções de utilização gráfica do simulador. De forma intuitiva é possível ver neste fluxograma geral como são as funções contidas na pasta *kernel* evocadas na construção da plataforma gráfica do simulador.


OSIP.m
Osip_f.m
position_mouse.m
mouseclickdown.m
Sobre um porto
Sobre um icon
Sobre a janela de Pré visualização
Sobre o esquematico
Com Ligação
Sim
Não
conect_port.m
mouseclickup.m
LigarIcon.m
keypressfunc.m
delete_connection.m
Apaga
Sim
keypressfunc.m
Não
Arrasta
Sim
IconMoveFnc.m
redraw_conections.m
Não
Click botão direito
Sim
CopyFunc.m
< nome >_f.m
Não
Duplo Click
Sim
Não
port_definitions
help_wind.m
'ok'
'cancel'
mousemovefnc.m
Flag Zoom activas
Sim
FuncZoomIn/Out
keypressfunc.m
Não
Click botão direito
Sim
generic_parameters.m
PasteFunc.m
Não
Sim
Duplo Click
Não
Osip_f.m
File
Edit
Simulation
Tools
Help
open
open_function.m
New Project
new_project.m
Import External Module
import_component.m
step_1.m
step_2.m
step_3_1.m
step_3_2.m
Load File Module
import_file_component.m
Save File Module
export_file_component.m
Save
Save & Exit
save_function.m
Save AS
save_as_function.m
Undo
undo_function.m
Copy
CopyFunc.m
Paste
PasteFunc.m
Run
run_simulation.m
branch_route.m
write_mfile.m
Set Parameters
generic_parameters.m
Zoom In
FuncZoomIn.m
Zoom Out
FuncZoomOut.m
Create Componente
creat_component_hierarchy.m
hierarchy1.m
creat_comp_in.m
creat_comp_out.m
creat_component_hierarchy2.m
About OSIP
about_osip.m