

LAB 1 - Sistema de Controle de Trem de Pouso

CES-65 - Projetos de Sistemas Embarcados

CE-235 - Sistemas Embarcados de Tempo Real

CE-230 - Qualidade, Confiabilidade e Segurança (*Safety*) de Software

CE-237 - Tópicos Avançados de Teste de Software

Prof. Dr. Adilson Marques da Cunha

Prof. Dr. Luiz Alberto Vieira Dias



OBJETIVOS

- Mostrar de maneira clara, pragmática e didática o desenvolvimento de um projeto, utilizando-se do software SCADE (*Safety-Critical Application Development Environment*) da empresa ANSYS/Esterel Technologies.
- Servir de base e fonte de apoio para a capacitação e o nivelamento das Turmas de Graduação e de Pós-Graduação do ITA, envolvendo Projetos de Sistemas Computadorizados que se utilizam do SCADE.
- Descrever os procedimentos para a construção de um Sistema Embarcado de Tempo Real de Trem de Pouso (*Landing Gear*) de uma aeronave fictícia.

CONTEXTO

Com intuito de prover auxílio técnico ao Ambiente Integrado de Ferramentas de Engenharia de Software Ajudada por Computador do SCADE, para apoiar os alunos integrantes de Disciplinas CES-65, CE-230, CE-235 e CE-237 do ITA, este exercício de Laboratório tem por objetivo servir de guia ou tutorial, propiciando as condições mínimas, necessárias e suficientes para o entendimento e a construção de um Sistema Embarcado de Tempo Real envolvendo o Trem de Pouso de uma aeronave fictícia, utilizando-se a Ferramenta SCADE.

Os procedimentos mostrados neste documento não têm a intenção de esgotar o assunto. Eles foram elaborados visando apenas aumentar a clareza e o pragmatismo do Exercício de Laboratório, propiciando sua implementação satisfatória.

Voltado para um estudo orientado a um projeto, este trabalho busca selecionar o que há de melhor nos principais guias de estudo dos Semestres anteriores das disciplinas de Projetos de Sistemas Embarcados no ITA, envolvendo diversas discussões em fóruns a respeito do assunto. Portanto, ele representa um produto do autor e o resultado de um esforço coletivo dos alunos das turmas dos anos anteriores.

Conforme apontado por Samoel Mirachi e Carlos Lopes Nunes Júnior, em 2013, o Projeto do Trem de Pouso cria um display de indicação de trem de pouso. Com ele, torna-se possível indicar se o Trem de Pouso encontra-se: baixado e travado; levantado e travado; ou em transição de um estado para outro; além de mostrar falha no Trem de Pouso, em cada um dos seus estados.

Apesar de não ser obrigatório, pois este material buscou ser o mais didático possível, é recomendável que o leitor tenha um conhecimento mínimo de algoritmos e de programação.

Espera-se que, com este guia inicial, a jornada de estudo e produção de cada aluno seja facilitada e otimizada em futuras e novas implementações neste e/ou nos próximos semestres no ITA.

*Nível de Estudo: Pré-intermediário

“Uma imagem vale mais que mil palavras”

(Confúcio)

Dicionário de Termos

Alguns termos utilizados neste projeto, são peculiares ao uso do SCADE ou à área em questão. Portanto, são listados abaixo, alguns termos que normalmente não aparecem na área comum de programação. São eles:

👉 **Cockpit Display System (CDS)** → Oferece a parte visível (e/ou audível) de um sistema cuja função seja de Interação Humano-Computador (IHC). Representa, basicamente, a tela que apresenta informações para o utilizador do sistema, podendo ser um componente de hardware físico ou até mesmo uma emulação de tela.

CDSs de um “Legacy 500”, aeronave executiva produzida pela EMBRAER.



Fonte: Portal da international-pilot

👉 **SCADE UAPage Creator/Display** → Módulo do SCADE responsável pelo desenvolvimento visual do projeto. Extensão padrão: “.sgfx”

👉 **SCADE Suite** → Módulo do Scade responsável pelo desenvolvimento da lógica do projeto. Extensão padrão: “.vsw”

👉 **Definition File (DF)** → Instancia de arquivo que irá definir a parte visual do projeto, ou seja o layout, disposições, tamanhos, cores dentre outros atributos de botões, textos, etc. A GUI (Graphical User Interface) é definida em binário. O DF é reconhecido pela extensão “.sgfx”, pode ser criado e editado no módulo do Scade Page Creator.

👉 **Widgets** → É um elemento (controle) de interação do usuário com o computador. Este elemento normalmente é disposto dentro de uma interface gráfica de usuário (GUI), com isso o usuário pode interagir de forma direta com o controle/elemento para ler, inserir ou editar informações de um aplicativo. As bibliotecas de Widgets vêm acompanhadas de botões, menus, contêineres,

Edit Boxes, listas, radioboxes, barra de progresso e de rolagem, entre inúmeros outros.

👉 **User Application (UA)** → Aplicativos do usuário/ Lógica do sistema.

👉 **WorkSpace** → Instancia de arquivo que irá definir dentro da área no Scade de desenvolvimento de programação, onde é inserida toda a lógica de desenvolvimento e visualização dos objetos, atributos e métodos, além das ferramentas para esse desenvolvimento. Reconhecido pela extensão “.vsw”, pode ser editável no módulo do Scade Suite.

👉 **ARINC 661** → É uma norma de padronização que visa a normalização, comunicação e definição dos elementos de um ou mais CDSs. Em outras palavras, busca fazer a interação e comunicação dos vários CDSs com a UA's do sistema. Entre as diversas aeronaves que utilizam este padrão podemos citar Airbus A380 e Boeing 787. No nosso caso também pode ser entendido como um protocolo de comunicação entre UA e CDS.

Sumário de Desenvolvimento

| | | |
|--------------------|-------------------------|-----------|
| 0 | Instalação do Scade | 5 |
| 1 | Desenvolvendo o visual | 6 |
| 2 | Desenvolvendo a lógica | 11 |
| 3 | Reportando documentação | 46 |
| Referências | | 50 |

0 Instalação do Scade

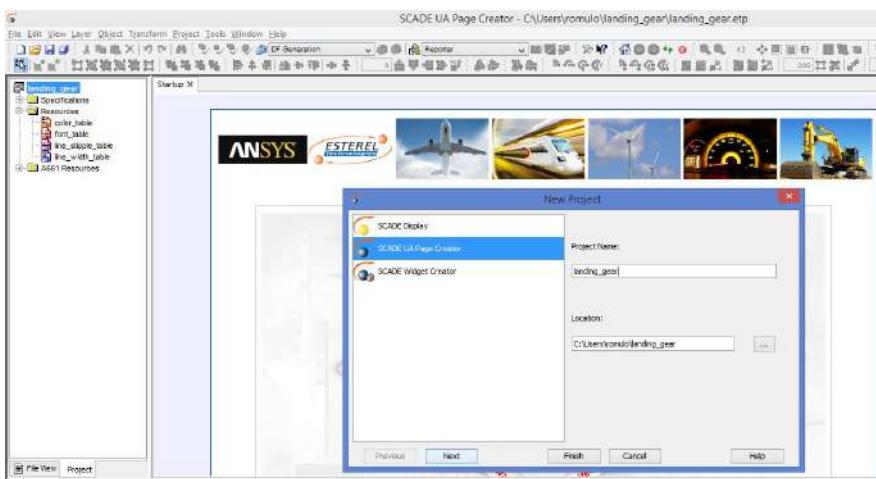
- 👉 Recomenda-se a plataforma Windows 8.1 para a instalação do Scade.
- 👉 Os procedimentos de instalação podem ser encontrados no Portal do **Projeto de STAGIHO-TR**, ou [clique](#) aqui para ser redirecionado diretamente para a secção referida.

1 Primeiro abra o SCADE UA Page Creator:

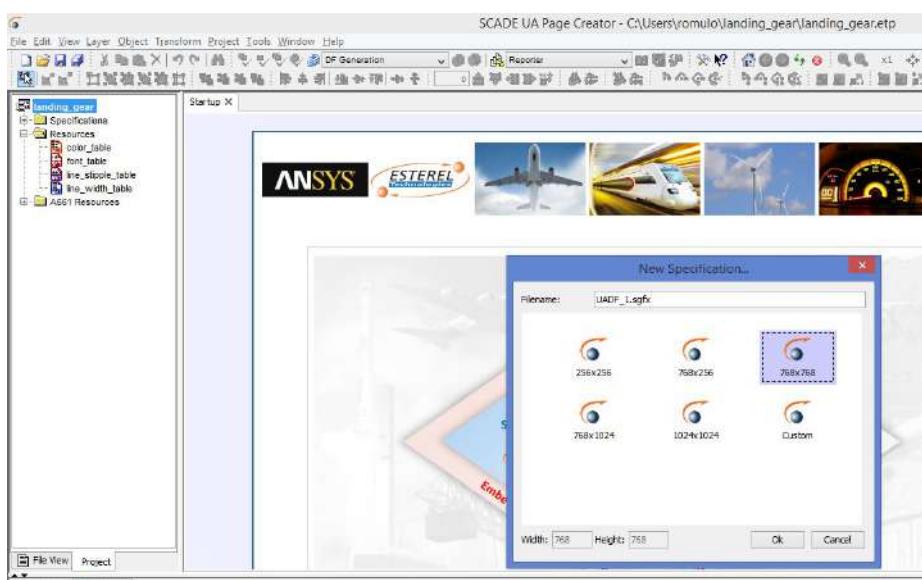


1.2 Crie um Projeto clicando em *File > New > Project*.

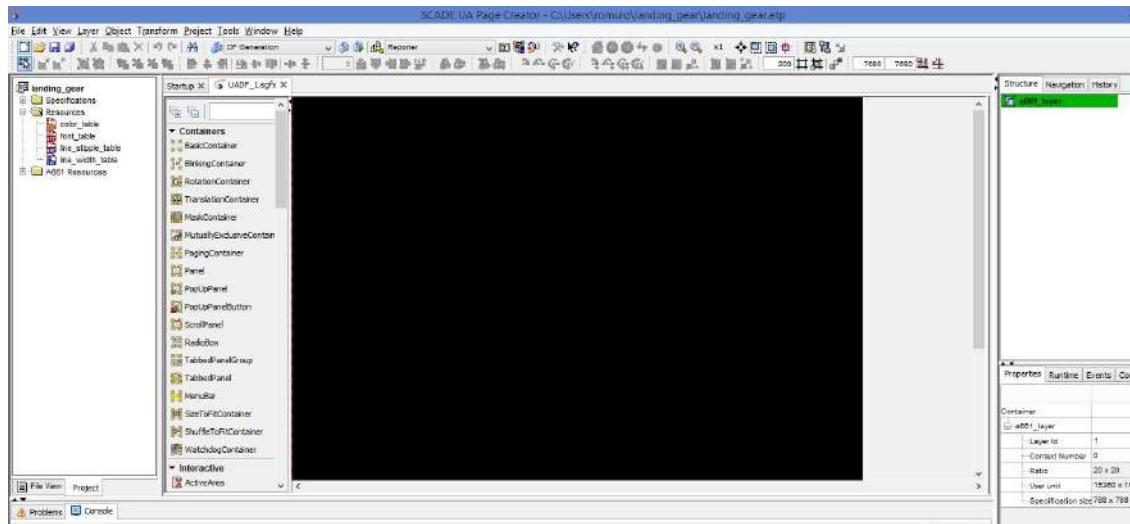
1.3 Selecione a opção *SCADE UA Page Creator*, e dê o nome *landing_gear* ao seu Projeto.



1.4 Agora crie um Definition File, clicando em *File>New>"Specification*, escolha a opção 768x768 DF dentro do menu Specification e pressione ok:

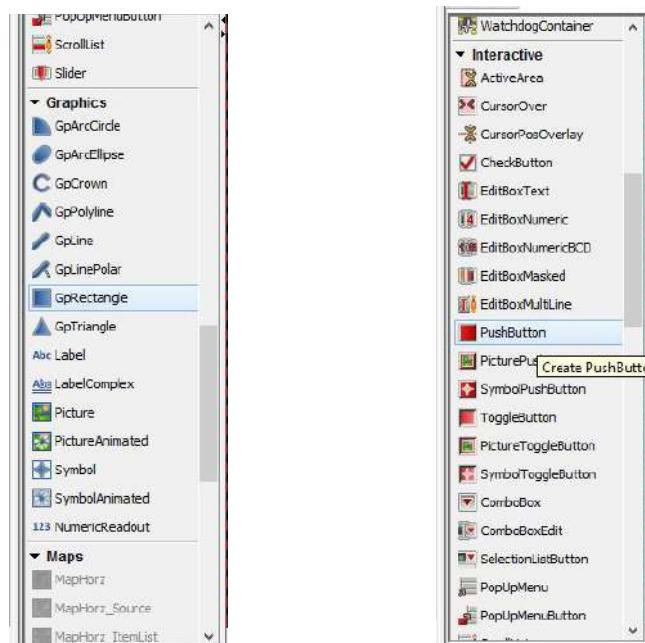


1.5 Agora, aparecerá a seguinte tela:

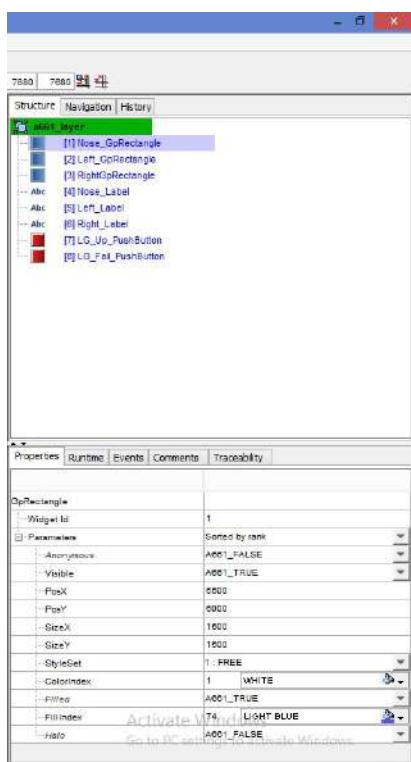


1.6 Para adicionar elementos ao display (área preta da figura anterior), apenas clique no desejado (objetos representados por ícones nos menus) e depois clique no display.

1.7 Agora, adicione 3 “GpRectangle”, 3 “Labels” e 2 “PushButtons”, não se preocupe com o posicionamento dos *widgets*, neste momento.



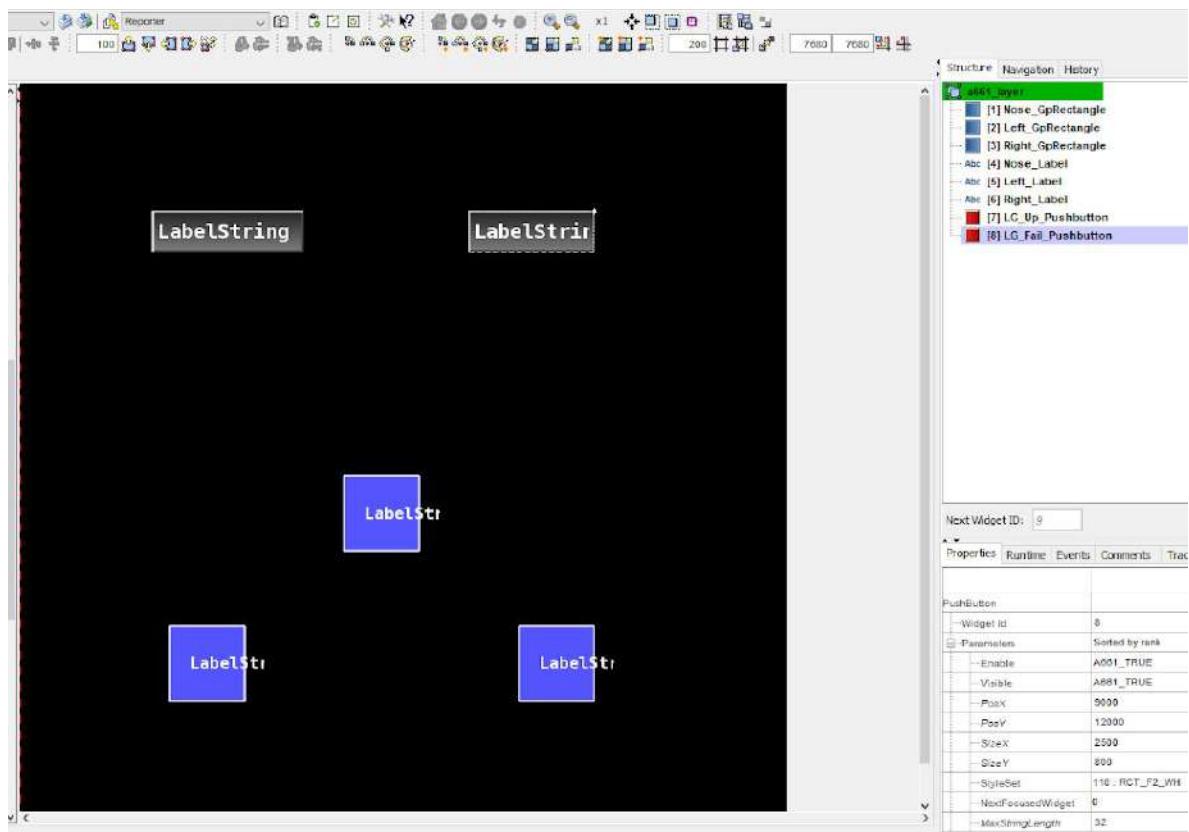
1.8 Agora, para editar a forma e a posição dos *widgets* de modo mais preciso, utilize o “*Properties box*”, a direita, no canto inferior. Para cada elemento criado na aba *Structure*, as propriedades listadas no passo seguinte devem ser editadas.



1.9 Os widgets devem ser nomeados e posicionados, de acordo com a tabela abaixo:

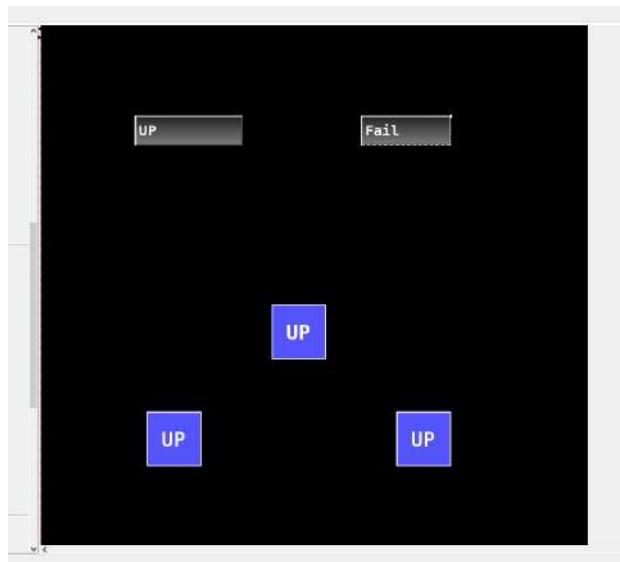
| Nome | Widget ID | PosX | PosY | SizeX | SizeY |
|--------------------|-----------|-------|-------|-------|-------|
| Nose_GpRectangle | 1 | 6500 | 6000 | 1500 | 1500 |
| Left_GpRectangle | 2 | 3000 | 3000 | 1500 | 1500 |
| Right_GpRectangle | 3 | 10000 | 3000 | 1500 | 1500 |
| Nose_Label | 4 | 6900 | 6000 | 1500 | 1500 |
| Left_Label | 5 | 3400 | 3000 | 1500 | 1500 |
| Right_Label | 6 | 10400 | 3000 | 1500 | 1500 |
| LG_Up_Pushbutton | 7 | 3000 | 12000 | 2500 | 800 |
| LG_Fail_Pushbutton | 8 | 9000 | 12000 | 2500 | 800 |

1.10 Ao final, seu display deverá ficar como na figura a seguir:

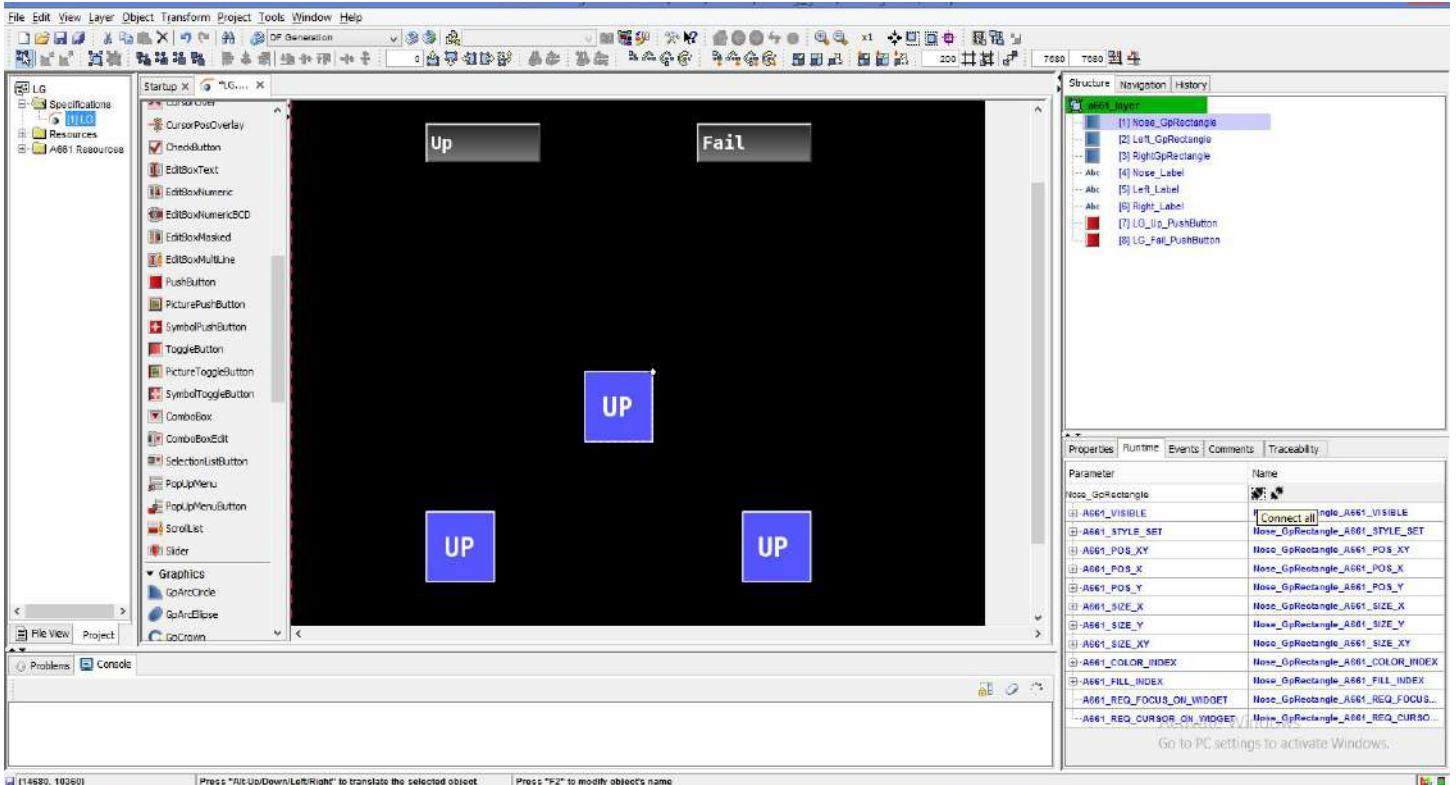


1.11 Agora, mude o texto dos labels para UP e deixe o tamanho da fonte como "F4" e do LG_UP_Pushbutton para UP, e o LG_Fail_Pushbutton para Fail.

O display deve ficar da seguinte forma:



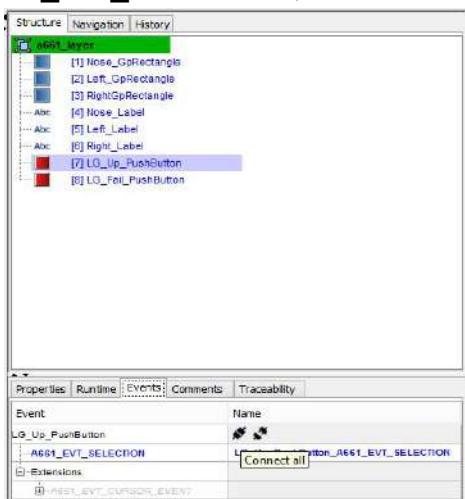
1.12 Ative as mensagens do barramento ARINC 661, para cada componente criado através do menu Runtime, clicando no botão “**Connect All**”, conforme mostrado abaixo:



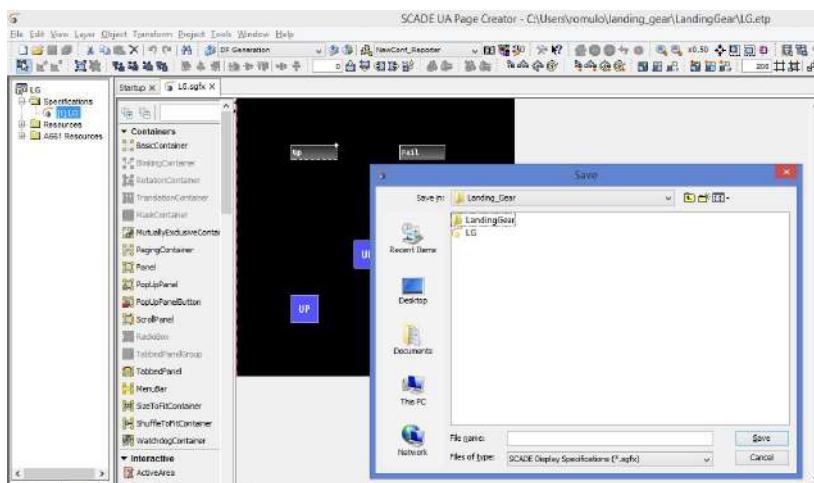
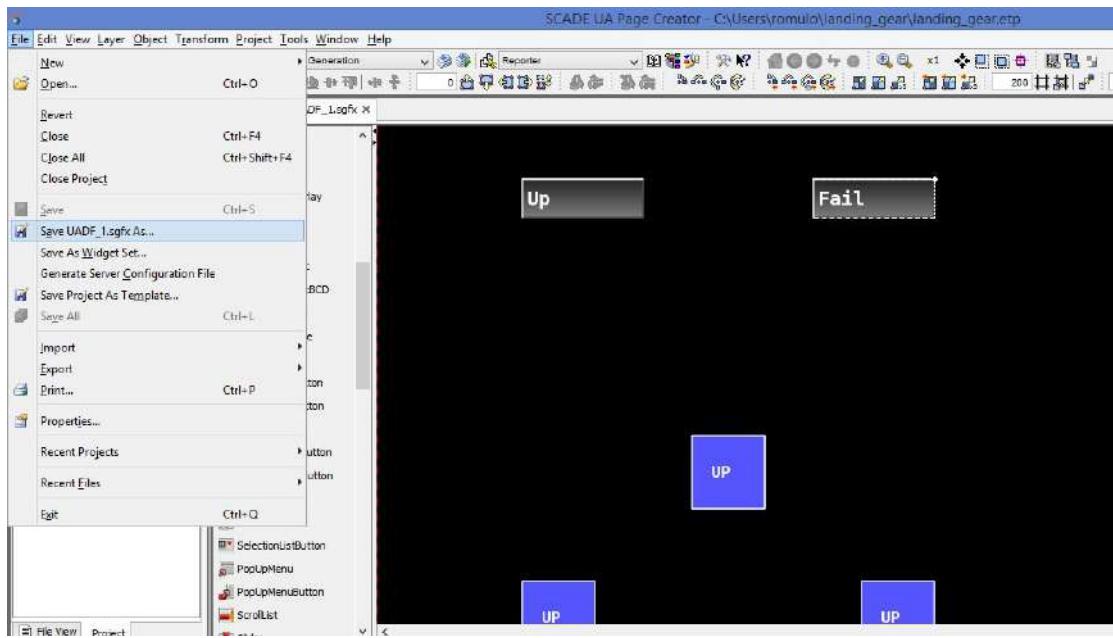
Obs: Lembre-se de repetir o processo para todos os componentes listados na sessão “*Structure*”. **Incluindo o objeto a661_layer.**

Obs2: No SCADE R18 a opção para fazer a conexão do objeto **a661_layer** pode não aparecer, para aparecer salve o projeto, feche-o e abra novamente, assim a opção para conectá-lo estará disponível.

1.13 Configure os eventos dos componentes *LG_Up_PushButton* e *LG_Fail_PushButton*, clicando na aba *Events* e *Connect All*.



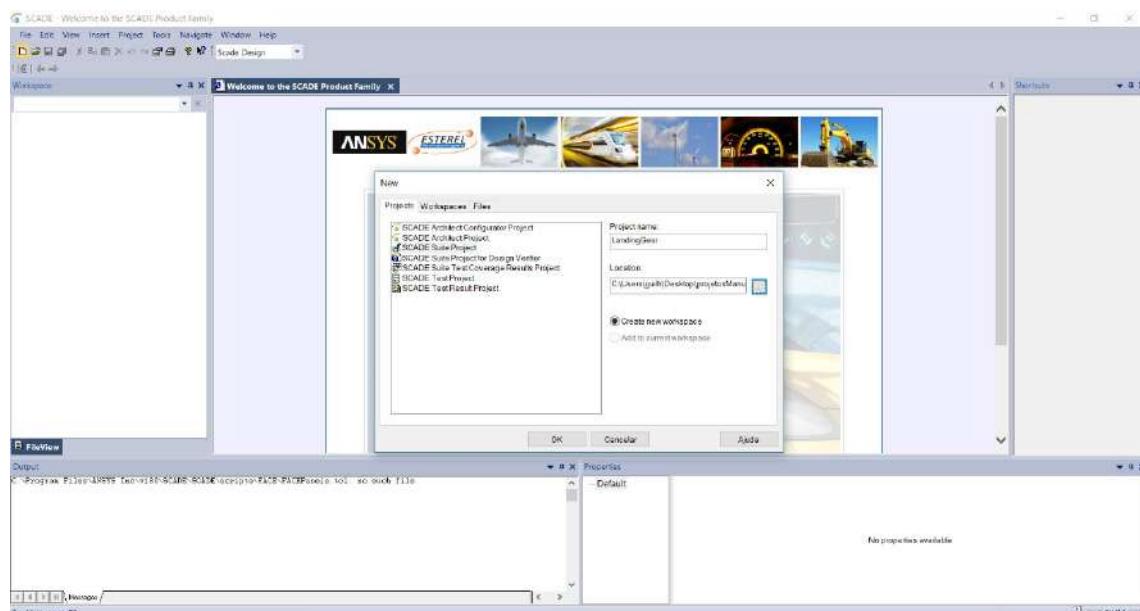
1.14 Agora, salve o seu **File (DF)** como “LG” (observe como ficará a extensão do arquivo .sgfx), dentro de uma pasta que se possa achar com facilidade.



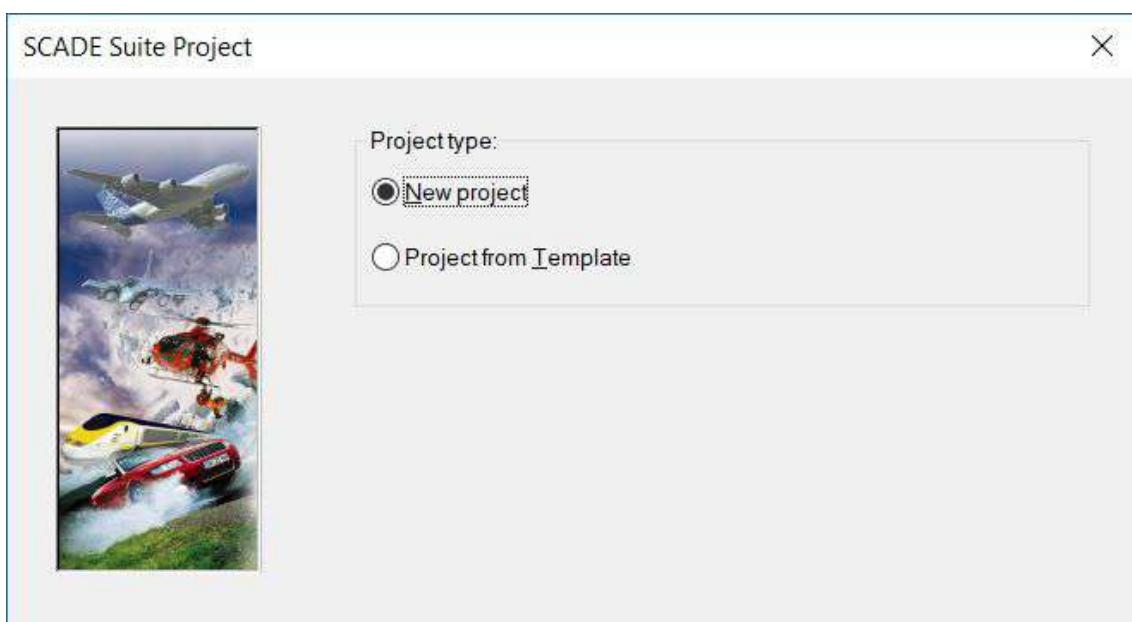
2 Agora, abra o SCADE Suite R19.2



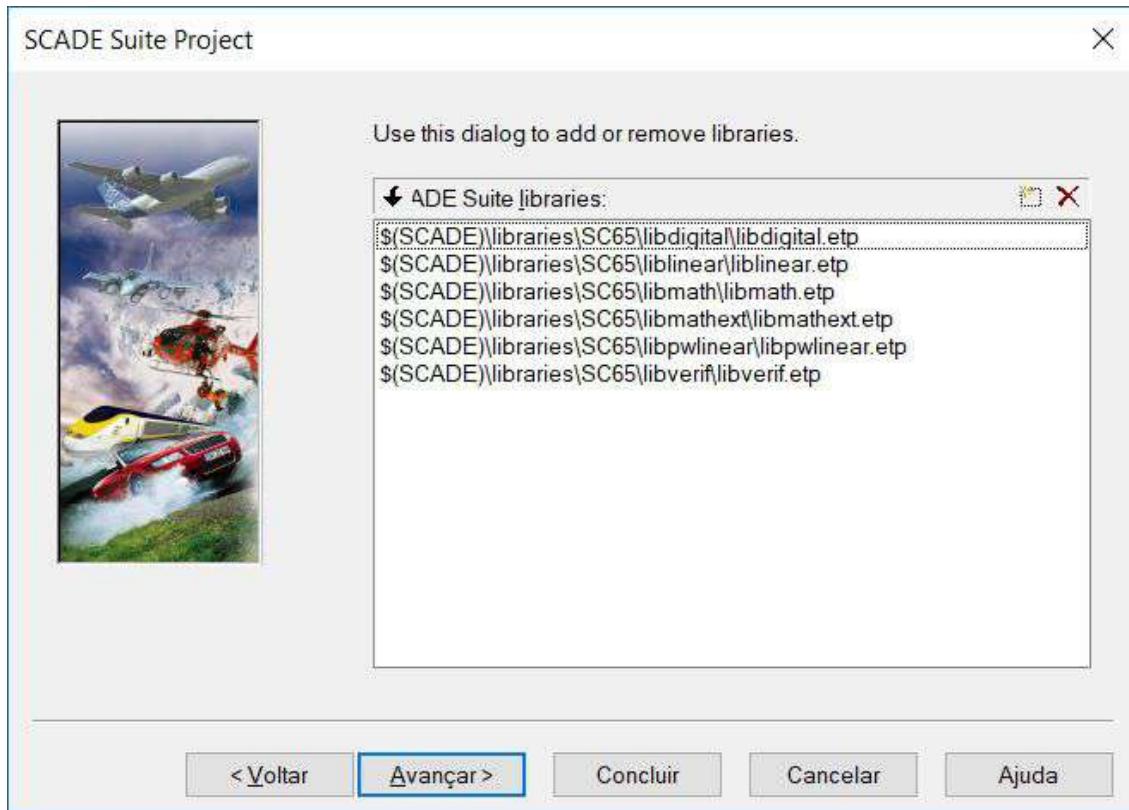
2.1 Agora, crie um Projeto chamado *LandingGear* e salve ele na mesma pasta do Definition File (DF).



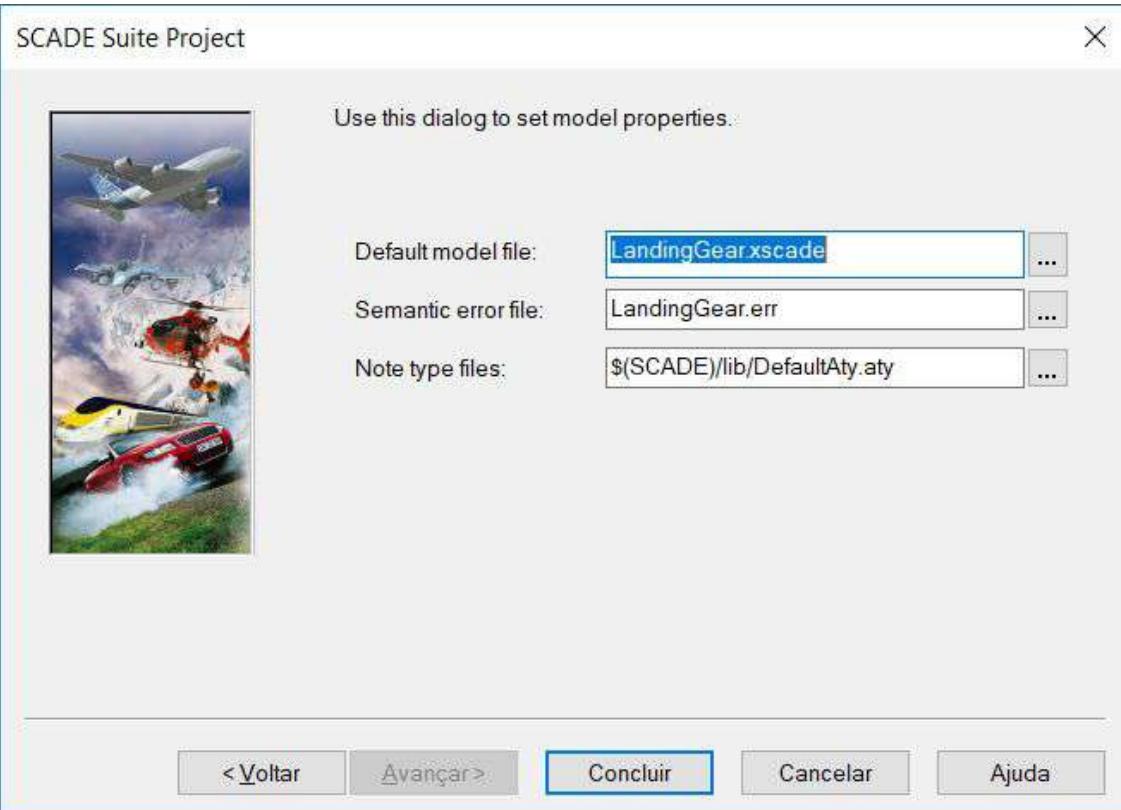
Passo 1



Passo 2

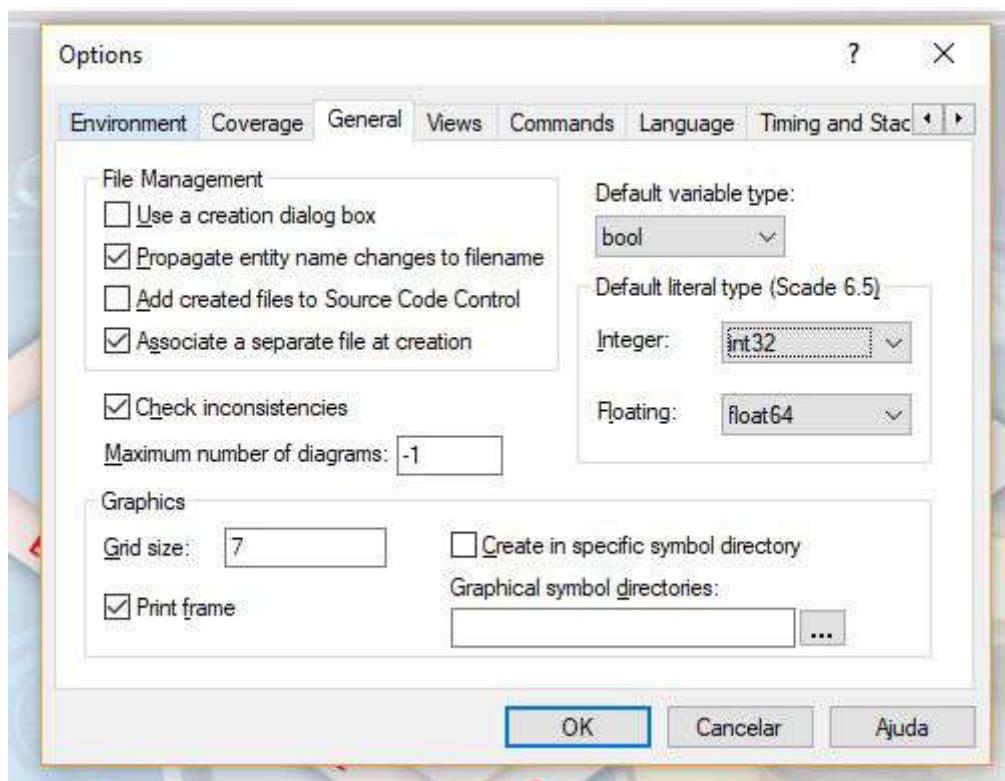


Passo 3

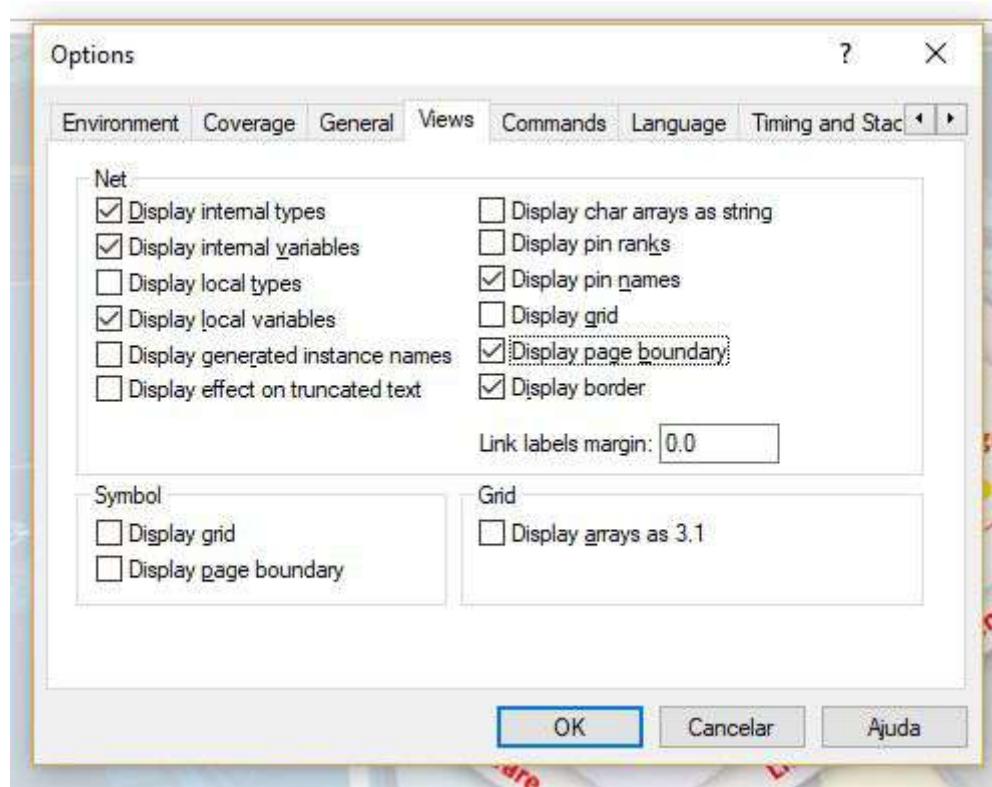


Passo 4

2.2 Agora, configure o SCADE Suite desta maneira: Vá no menu Tools> Options e clique na aba General, nesta aba, marque as seguintes opções:

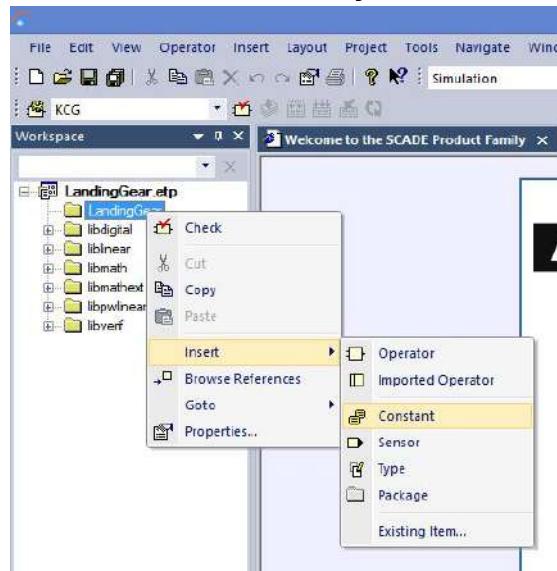


2.3 Agora clique na aba Views e marque as seguintes opções:

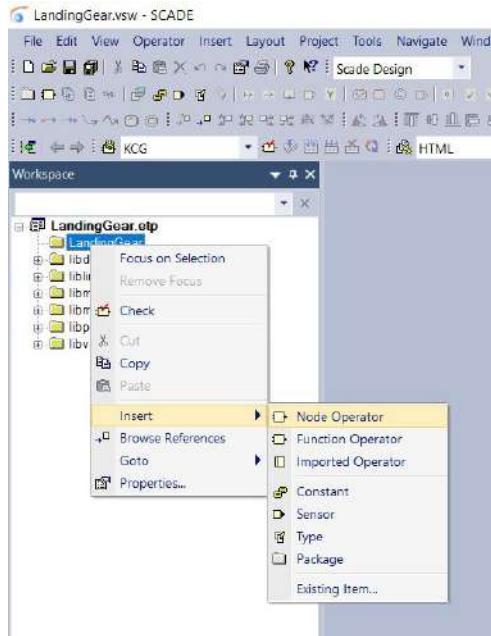


2.4 Agora serão explicadas algumas funcionalidades básicas do SCADE Suite.

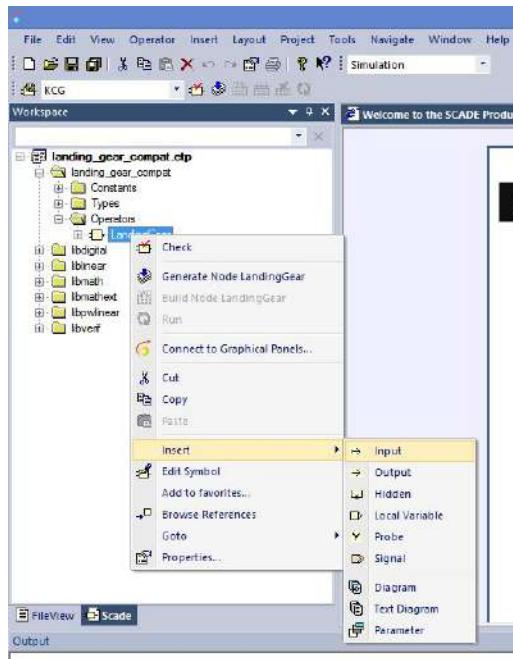
Constantes: São valores de certo tipo de dado que podem ser atribuídos a outros elementos. A criação de uma Constante no Scade é feita desta maneira:



Operadores: são utilizados para construir e executar a lógica de seu programa. A criação de um operador no Scade é feita desta maneira:

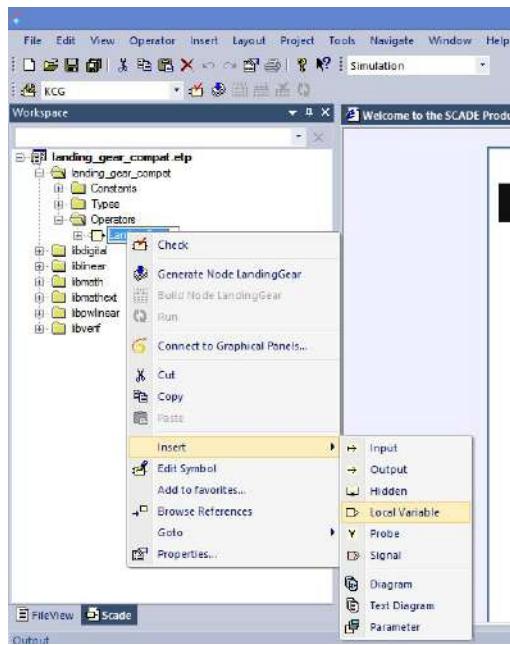


Interfaces Inputs ou Outputs: Podem ser utilizadas para se definir os valores de uma variável ao receber/enviar o valor de uma ação. A criação de uma interface é feita da seguinte maneira no SCADE:

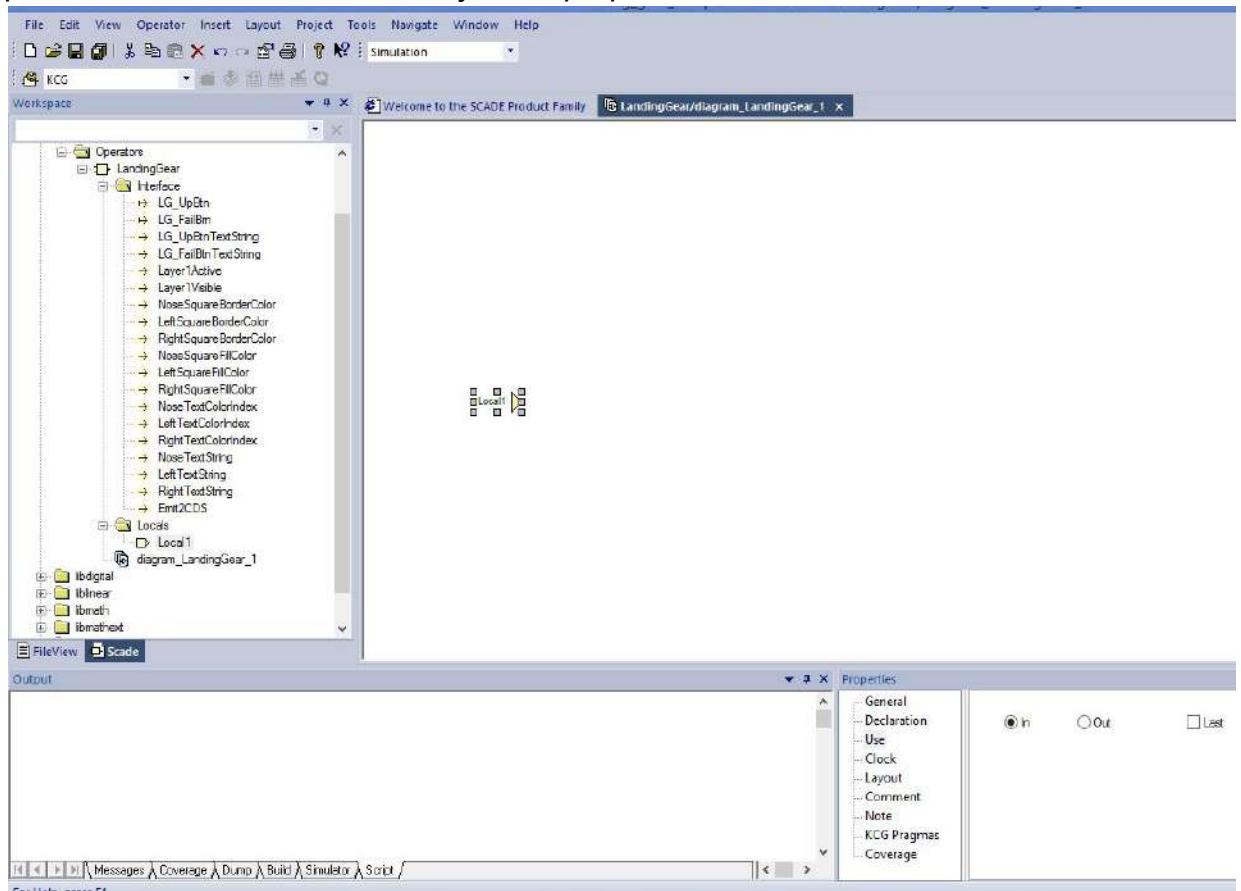


Variáveis locais: São utilizadas para armazenar valores.

A criação de uma variável local é feita desta maneira no Scade:

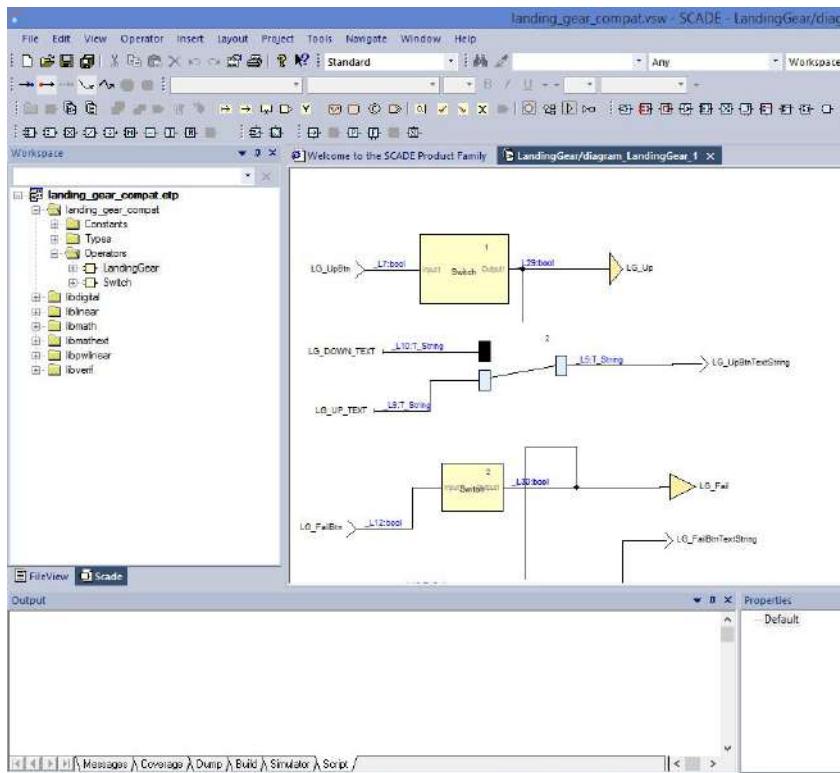


Obs: uma variável pode ser usada tanto como input quanto como output e pode-se escolher o uso dela na janela “properties”.



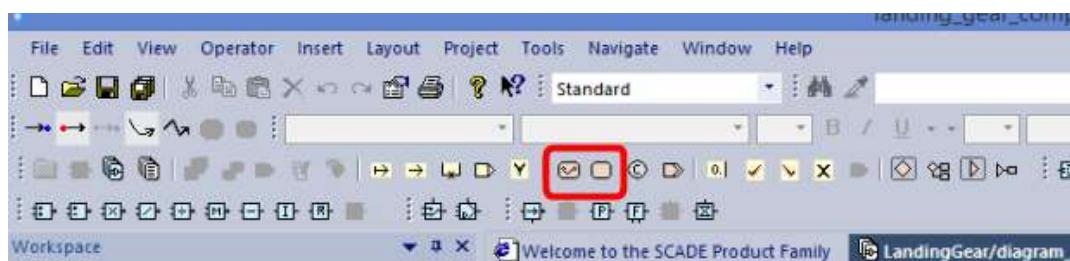
Workspace: Clicando duas vezes em um Operador, surgirá uma tela branca que é utilizada para montar a lógica do programa como um “círcuito”(quem já projetou algo em eletrônica vai se sentir familiarizado o.O xD) arrastando vários elementos como variáveis, constantes, Máquinas de estado, Inputs, outputs e até outros operadores.

Exemplo de workspace:



Máquinas de estado e estados: Uma máquina de estado é utilizada para fazer a transição de uma ação para outra utilizando estados que pulam de ativos para desativados dependendo da ação, um estado possui um workspace próprio que é utilizado para inserir lógica assim como no Operador.

A criação de uma Máquina de estado e de um estado é feita clicando nestes ícones:



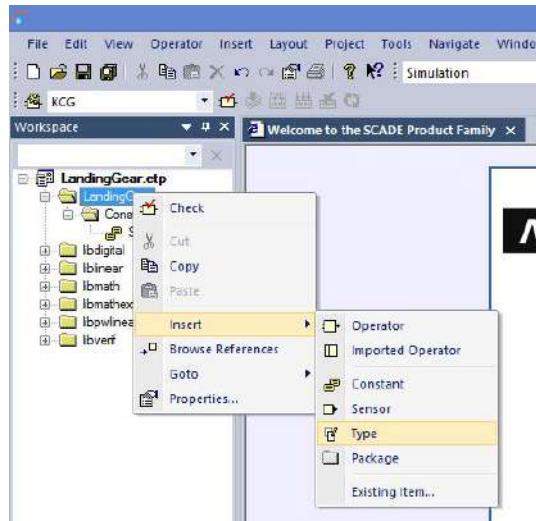
Operadores lógicos: Os operadores lógicos como IF, FOLLOWED BY, NOT etc. Estão incluídos no SCADE e podem ser utilizados no workspace, clicando conforme cada um deles, nesta área (As barras disponíveis podem ser configuradas com o clique do botão direito na área abaixo):



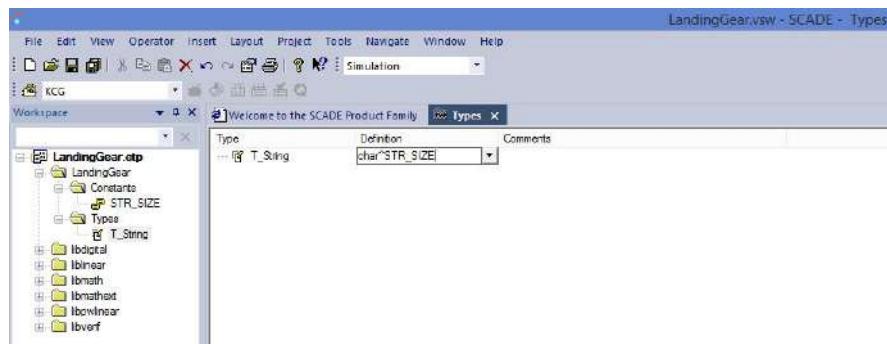
2.5 Agora, crie uma constante chamada STR_SIZE do tipo uint16 e com o valor 5 (para editar o valor e o tipo clique duas vezes sobre a constante recem criada)

| | |
|-------------|-------------------|
| Name: | STR_SIZE |
| Path: | STR_SIZE/ |
| Rename: | LandingGear.scdce |
| Visibility: | Public |

2.6 Agora crie um novo tipo chamado “T_String” clicando como segue a figura:



2.7 Agora edite o tipo criado e escreva e aperte a tecla ‘enter’:



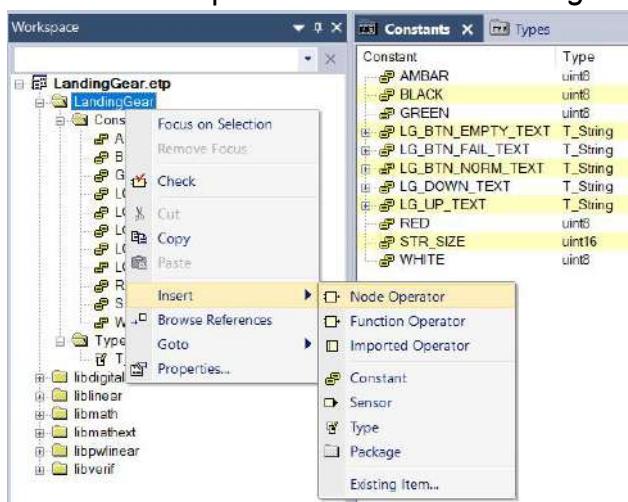
Desta maneira, o tipo criado se torna um array do tipo char de tamanho STR_SIZE

2.8 Agora, crie as seguintes constantes:

| Constant | Type | Value | Comments |
|-------------------|----------|---------------------------|----------|
| AMBAR | uint8 | 31 | |
| BLACK | uint8 | 0 | |
| GREEN | uint8 | 51 | |
| LG_BTN_EMPTY_TEXT | T_String | [' ', ' ', ' ', ' ', ' '] | |
| LG_BTN_FAIL_TEXT | T_String | ['F', 'A', 'I', 'L', ' '] | |
| LG_BTN_NORM_TEXT | T_String | ['N', 'O', 'R', 'M', ' '] | |
| LG_DOWN_TEXT | T_String | ['D', 'N', ' ', ' ', ' '] | |
| LG_UP_TEXT | T_String | ['U', 'P', ' ', ' ', ' '] | |
| RED | uint8 | 21 | |
| STR_SIZE | uint16 | 5 | |
| WHITE | uint8 | 1 | |

Obs: Não é necessário criar o STR_SIZE novamente.

2.9 Crie um Operador chamado *LandingGear*:

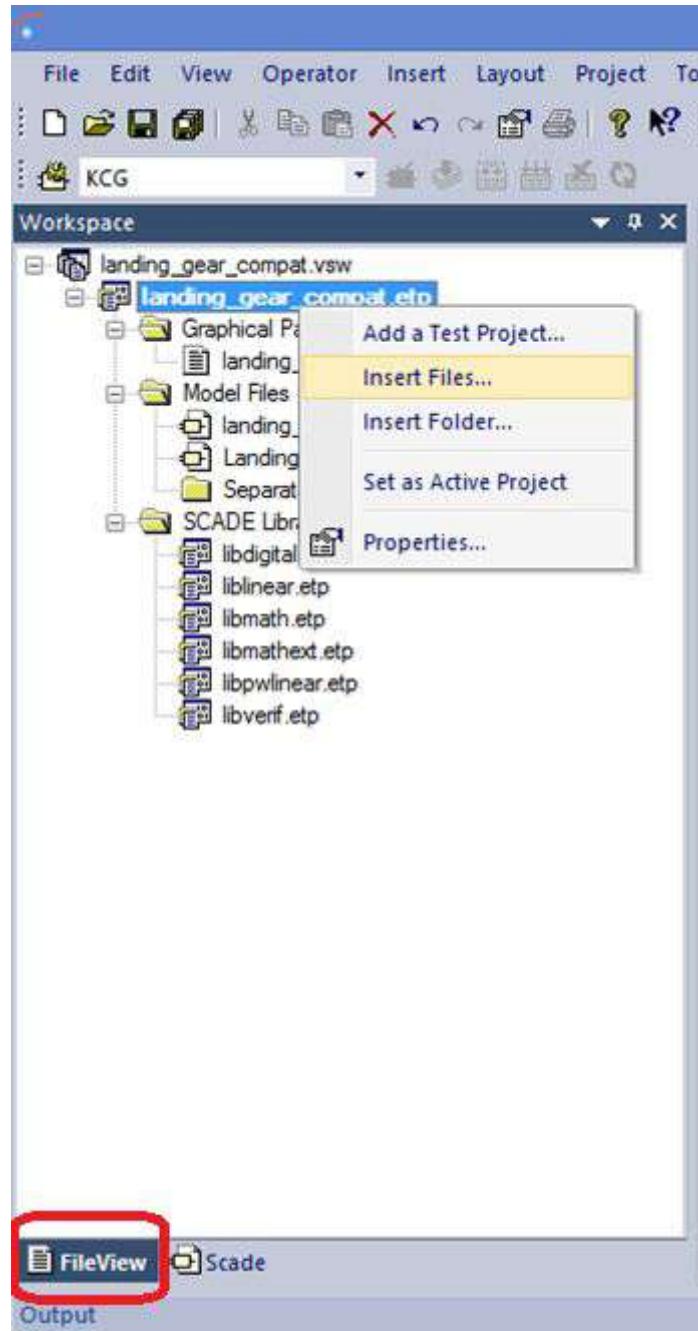


OBS: Para criar inputs e outputs, clique com o botão direito no operador e vá para *insert input/output*. O mesmo vale para as variáveis locais.

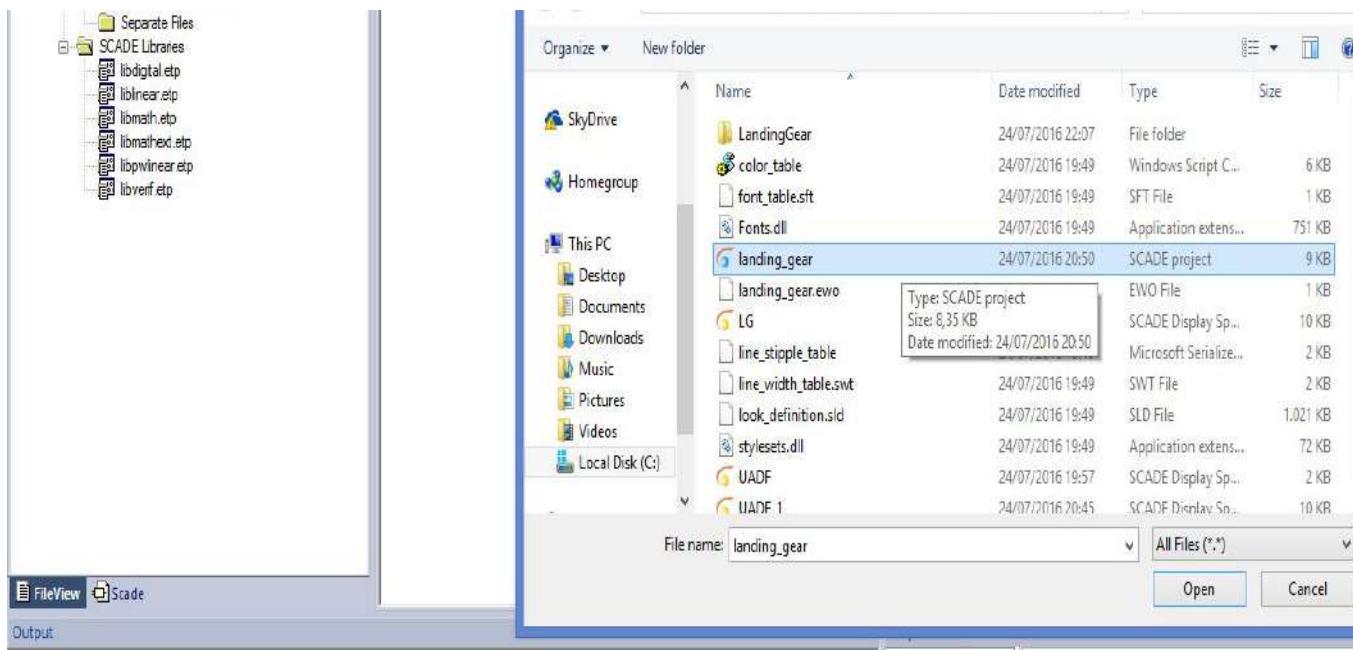
2.10 Agora, crie as interfaces inputs e outputs especificadas na tabela a seguir

| Name | Kind | Type |
|------------------------|--------|----------|
| LG_UpBtn | Input | bool |
| LG_FailBtn | Input | bool |
| LG_UpBtnTextString | Output | T_String |
| LG_FailBtnTextString | Output | T_String |
| Layer1Active | Output | bool |
| Layer1Visible | Output | bool |
| NoseSquareBorderColor | Output | uint8 |
| LeftSquareBorderColor | Output | uint8 |
| RightSquareBorderColor | Output | uint8 |
| NoseSquareFillColor | Output | uint8 |
| LeftSquareFillColor | Output | uint8 |
| RightSquareFillColor | Output | uint8 |
| NoseTextColorIndex | Output | uint8 |
| LeftTextColorIndex | Output | uint8 |
| RightTextColorIndex | Output | uint8 |
| NoseTextString | Output | T_String |
| LeftTextString | Output | T_String |
| RightTextString | Output | T_String |
| StringSize | Output | uint16 |
| Emit2CDS | Output | bool |

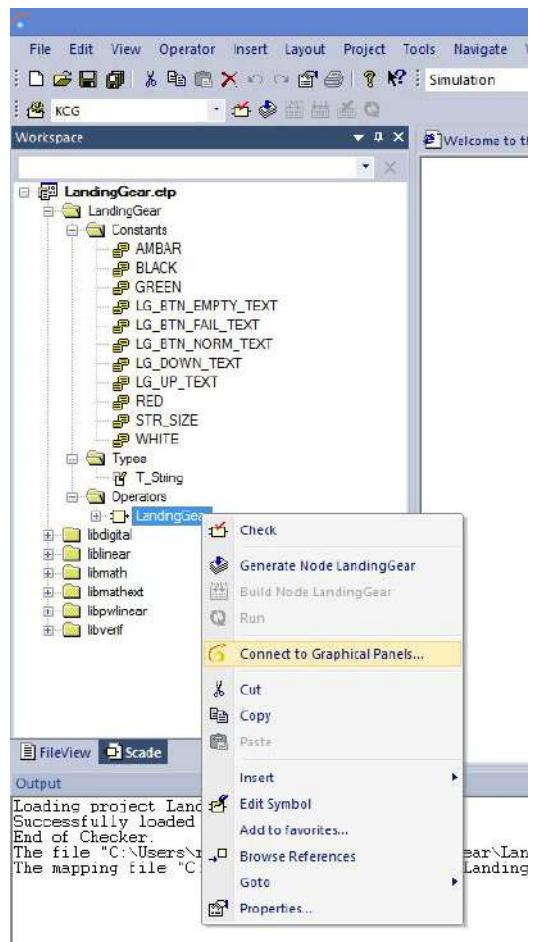
2.11 Agora importe o Definition File (que foi criado anteriormente) clicando na aba “File view” e com o botão direito no arquivo “LandingGear.etc” e indo em “insert files” escolha o arquivo Definition file que deve ter a extensão “*.etc” (como já mostrado anteriormente sobre o tipo de extensão do DF que é o “*.sgfx”):



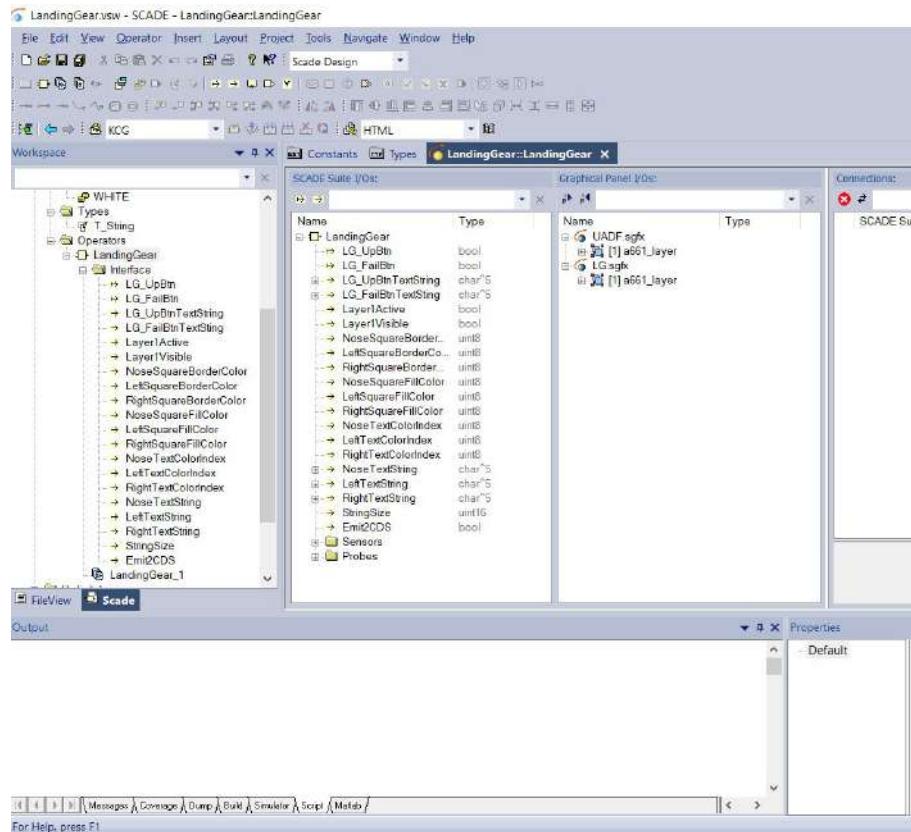
2.12 Escolha o Projeto do Definition File desenvolvido nos passos de 1.1 a 13.



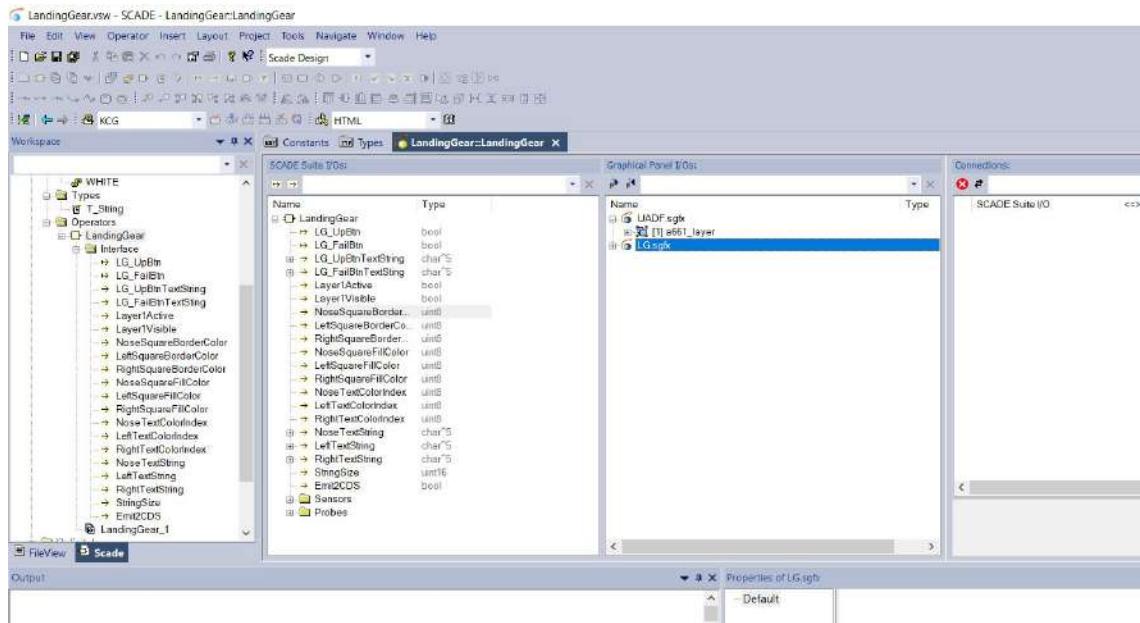
2.13 Agora, realize a conexão do operador *LandingGear* com o ambiente gráfico do Definition File, clique na aba SCADE e, em seguida, clique com o botão direito no operador *LandingGear* e, depois, vá para a opção “Connect to graphical panels”



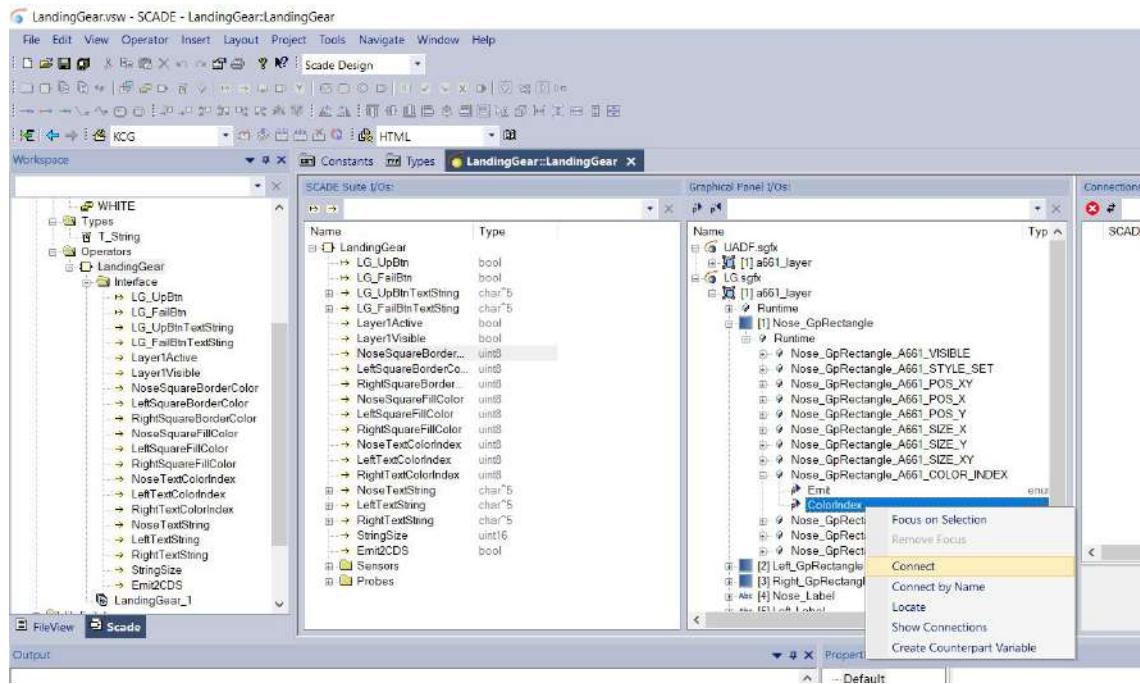
A tela abaixo deverá aparecer:



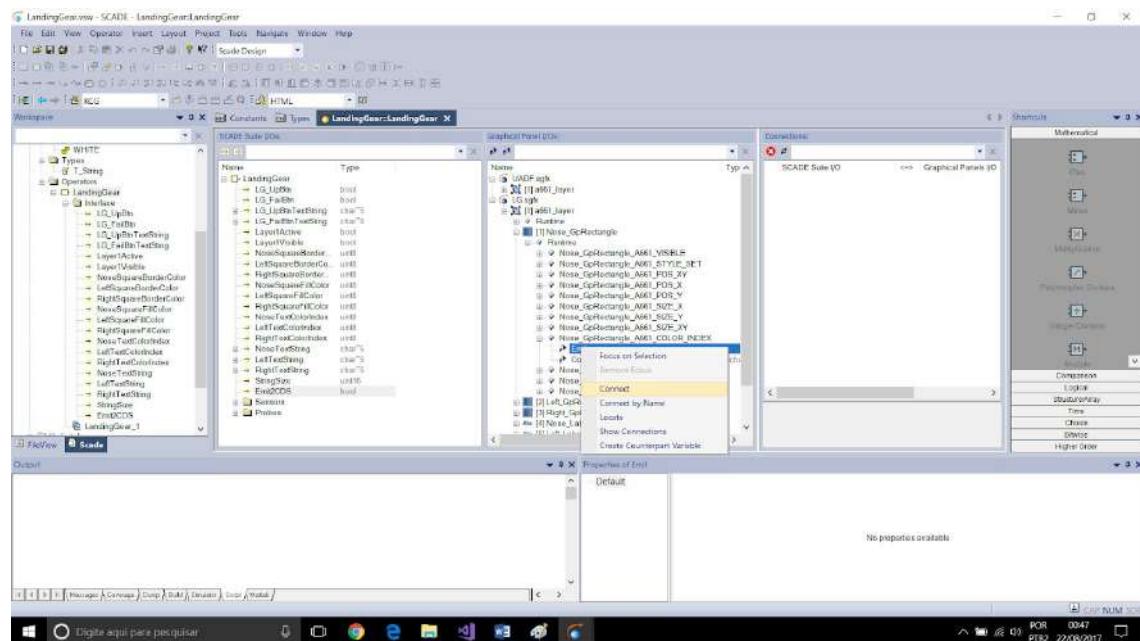
2.14 Na tela aberta, será feita a associação entre os inputs e outputs com as interface gráfica. Para fazer uma conexão, clique em uma I/O (Ex: NoseSquareBorderColor) e clique em um dos valores dos widgets (Ex: Nose_GpRectangle_A661_COLOR_INDEX). Observe o exemplo abaixo:

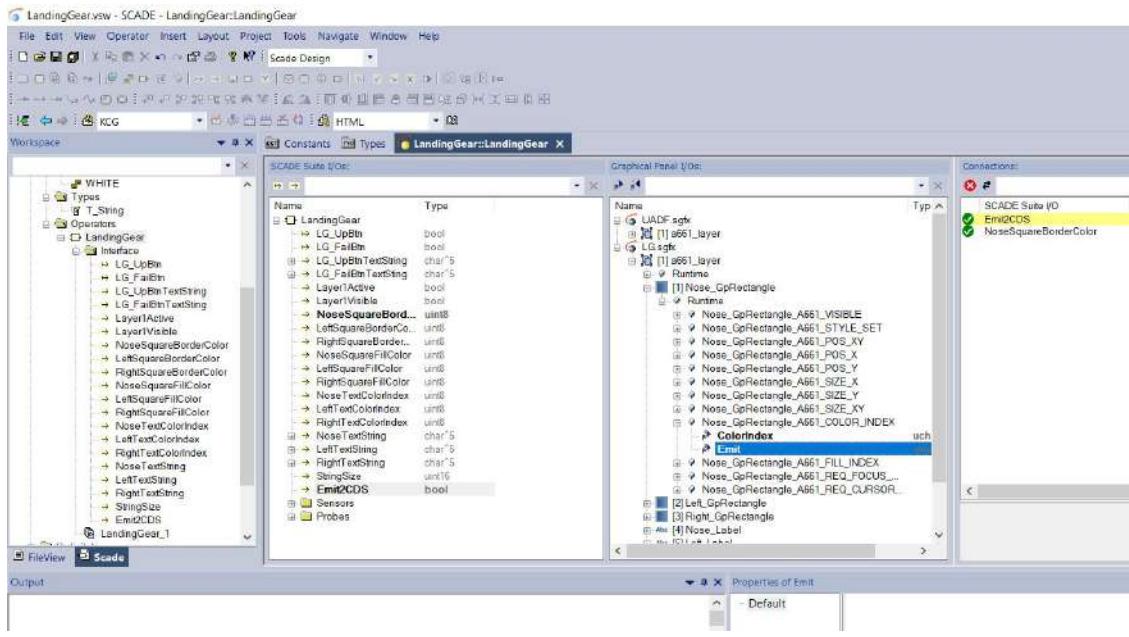


Selecionando o I/O e o valor de um Widget



Conectando uma I/O a um valor de um Widget





Obs: Observe que somente quando fazemos a união de todos os elementos de I/O correspondente a sinalização passará de incompleta (representado por uma exclamação “!”) para completar (um sinal de checado/checked “✓”)

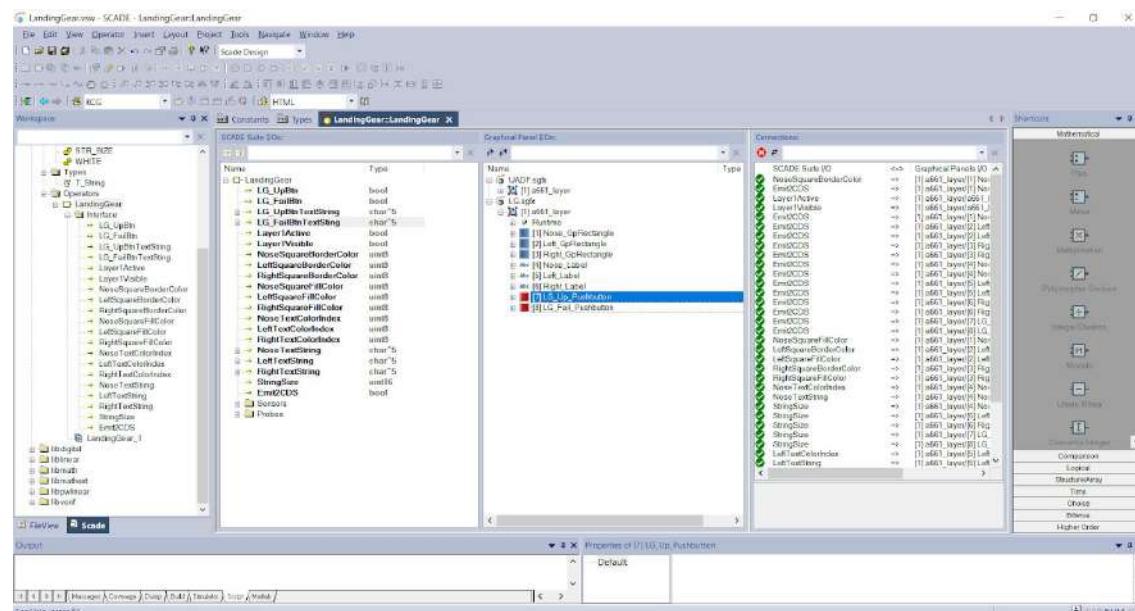
2.15 Agora, faça as conexões determinadas na tabela ou imagem abaixo:

| SCADE Suite Signal | Widget | A661 Property |
|------------------------|--------------------|-----------------------------|
| Layer1Active | Layer#1 | A661_REQ_LAYER_ACTIVE |
| Layer1Visible | | A661_REQ_LAYER_VISIBLE |
| NoseSquareBorderColor | Nose_GpRectangle | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| NoseSquareFillColor | | A661_FILL_INDEX.FillIndex |
| Emit2CDS | | A661_FILL_INDEX.Emit |
| LeftSquareBorderColor | Left_GpRectangle | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| LeftSquareFillColor | | A661_FILL_INDEX.FillIndex |
| Emit2CDS | | A661_FILL_INDEX.Emit |
| RightSquareBorderColor | Right_GpRectangle | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| RightSquareFillColor | | A661_FILL_INDEX.FillIndex |
| Emit2CDS | | A661_FILL_INDEX.Emit |
| NoseTextColorIndex | Nose_Label | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| NoseTextString | | A661_STRING.LabelString |
| StringSize | | A661_STRING.String_size |
| Emit2CDS | | A661_STRING.Emit |
| LeftTextColorIndex | Left_Label | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| LeftTextString | | A661_STRING.LabelString |
| StringSize | | A661_STRING.String_size |
| Emit2CDS | | A661_STRING.Emit |
| RightTextColorIndex | Right_Label | A661_COLOR_INDEX.ColorIndex |
| Emit2CDS | | A661_COLOR_INDEX.Emit |
| RightTextString | | A661_STRING.LabelString |
| StringSize | | A661_STRING.String_size |
| Emit2CDS | | A661_STRING.Emit |
| LG_UpBtn | LG_Up_PushButton | A661_EVT_SELECTION.Notify |
| LG_UpBtnTextString | | A661_STRING.LabelString |
| StringSize | | A661_STRING.String_size |
| Emit2CDS | | A661_STRING.Emit |
| LG_FailBtn | LG_Fail_PushButton | A661_EVT_SELECTION.Notify |
| LG_FailBtnTextString | | A661_STRING.LabelString |
| StringSize | | A661_STRING.String_size |
| Emit2CDS | | A661_STRING.Emit |

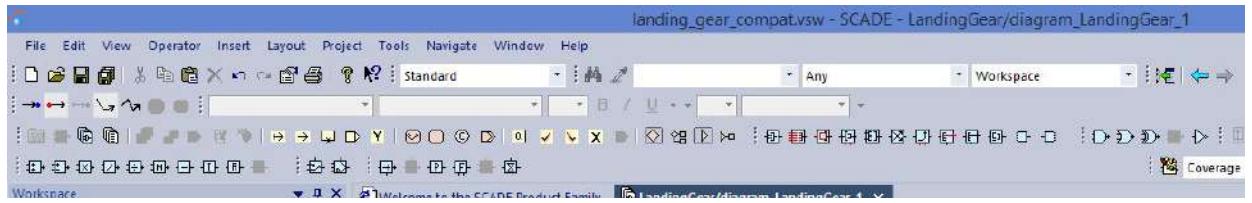
Verifique as conexões na imagem abaixo:

| SCADE Suite I/O | \leftrightarrow | Graphical Panels I/O |
|------------------------|-------------------|--|
| NoseSquareBorderColor | => | [1] a661_layer/[1] Nose_GpRectangle/Nose_GpRectangle_A661_COLOR_INDEX/ColorIndex |
| Layer1Active | => | [1] a661_layer/a661_layer_A661_REQ_LAYER_ACTIVE/Emit |
| Layer1Visible | => | [1] a661_layer/a661_layer_A661_REQ_LAYER_VISIBLE/Emit |
| Emit2CDS | => | [1] a661_layer/[1] Nose_GpRectangle/Nose_GpRectangle_A661_COLOR_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[1] Nose_GpRectangle/Nose_GpRectangle_A661_FILL_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[2] Left_GpRectangle/Left_GpRectangle_A661_COLOR_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[2] Left_GpRectangle/Left_GpRectangle_A661_FILL_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[3] RightGpRectangle/RightGpRectangle_A661_COLOR_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[3] RightGpRectangle/RightGpRectangle_A661_FILL_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[4] Nose_Label/Nose_Label_A661_COLOR_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[5] Left_Label/Left_Label_A661_STRING/Emit |
| Emit2CDS | => | [1] a661_layer/[6] Right_Label/Right_Label_A661_COLOR_INDEX/Emit |
| Emit2CDS | => | [1] a661_layer/[6] Right_Label/Right_Label_A661_STRING/Emit |
| Emit2CDS | => | [1] a661_layer/[7] LG_Up_PushButton/LG_Up_PushButton_A661_STRING/Emit |
| Emit2CDS | => | [1] a661_layer/[8] LG_Fail_PushButton/LG_Fail_PushButton_A661_STRING/Emit |
| NoseSquareFillColor | => | [1] a661_layer/[1] Nose_GpRectangle/Nose_GpRectangle_A661_FILL_INDEX/FillIndex |
| LeftSquareBorderColor | => | [1] a661_layer/[2] Left_GpRectangle/Left_GpRectangle_A661_COLOR_INDEX/ColorIndex |
| LeftSquareFillColor | => | [1] a661_layer/[2] Left_GpRectangle/Left_GpRectangle_A661_FILL_INDEX/FillIndex |
| RightSquareBorderColor | => | [1] a661_layer/[3] RightGpRectangle/RightGpRectangle_A661_COLOR_INDEX/ColorIndex |
| RightSquareFillColor | => | [1] a661_layer/[3] RightGpRectangle/RightGpRectangle_A661_FILL_INDEX/FillIndex |
| NoseTextColorIndex | => | [1] a661_layer/[4] Nose_Label/Nose_Label_A661_COLOR_INDEX/ColorIndex |
| NoseTextString | => | [1] a661_layer/[4] Nose_Label/Nose_Label_A661_STRING/LabelString |
| StringSize | => | [1] a661_layer/[4] Nose_Label/Nose_Label_A661_STRING/String_size |
| StringSize | => | [1] a661_layer/[5] Left_Label/Left_Label_A661_STRING/String_size |
| StringSize | => | [1] a661_layer/[6] Right_Label/Right_Label_A661_STRING/String_size |
| StringSize | => | [1] a661_layer/[7] LG_Up_PushButton/LG_Up_PushButton_A661_STRING/String_size |
| StringSize | => | [1] a661_layer/[8] LG_Fail_PushButton/LG_Fail_PushButton_A661_STRING/String_size |
| LeftTextColorIndex | => | [1] a661_layer/[5] Left_Label/Left_Label_A661_COLOR_INDEX/ColorIndex |
| LeftTextString | => | [1] a661_layer/[5] Left_Label/Left_Label_A661_STRING/LabelString |
| RightTextColorIndex | => | [1] a661_layer/[6] Right_Label/Right_Label_A661_COLOR_INDEX/ColorIndex |
| RightTextString | => | [1] a661_layer/[6] Right_Label/Right_Label_A661_STRING/LabelString |
| LG_UpBtnTextString | => | [1] a661_layer/[7] LG_Up_PushButton/LG_Up_PushButton_A661_STRING/LabelString |
| LG_UpBtn | \Leftrightarrow | [1] a661_layer/[7] LG_Up_PushButton/LG_Up_PushButton_A661_EVT_SELECTION/Notify |
| LG_FailBtn | \Leftrightarrow | [1] a661_layer/[8] LG_Fail_PushButton/LG_Fail_PushButton_A661_EVT_SELECTION/Notify |
| LG_FailBtnTextString | => | [1] a661_layer/[8] LG_Fail_PushButton/LG_Fail_PushButton_A661_STRING/LabelString |

2.16 O resultado do passo anterior deverá ser exatamente como mostrado na tela a seguir.



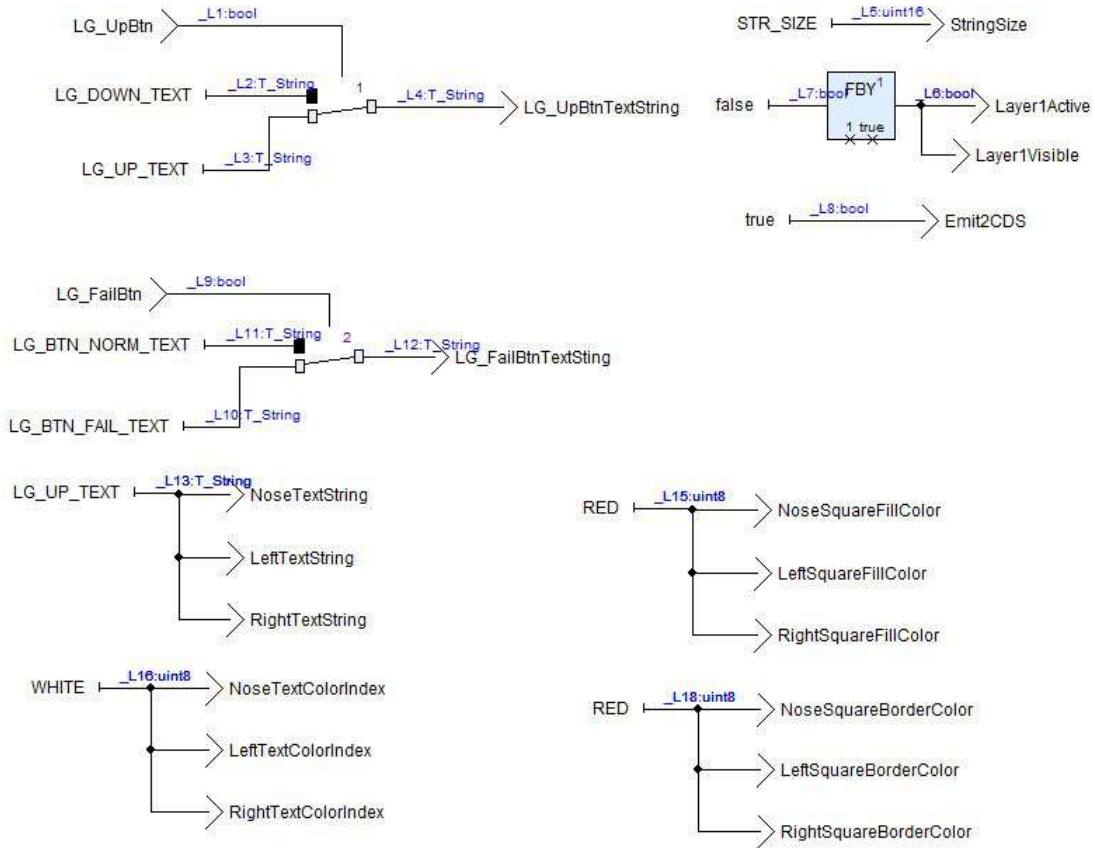
2.17 Configure as barras de ferramentas do SCADE, ativando as opções *Choice*, *Structure/Array*, *Logical*, *Time* e *Create* e a interface deverá ser semelhante a mostrada na tela a seguir:



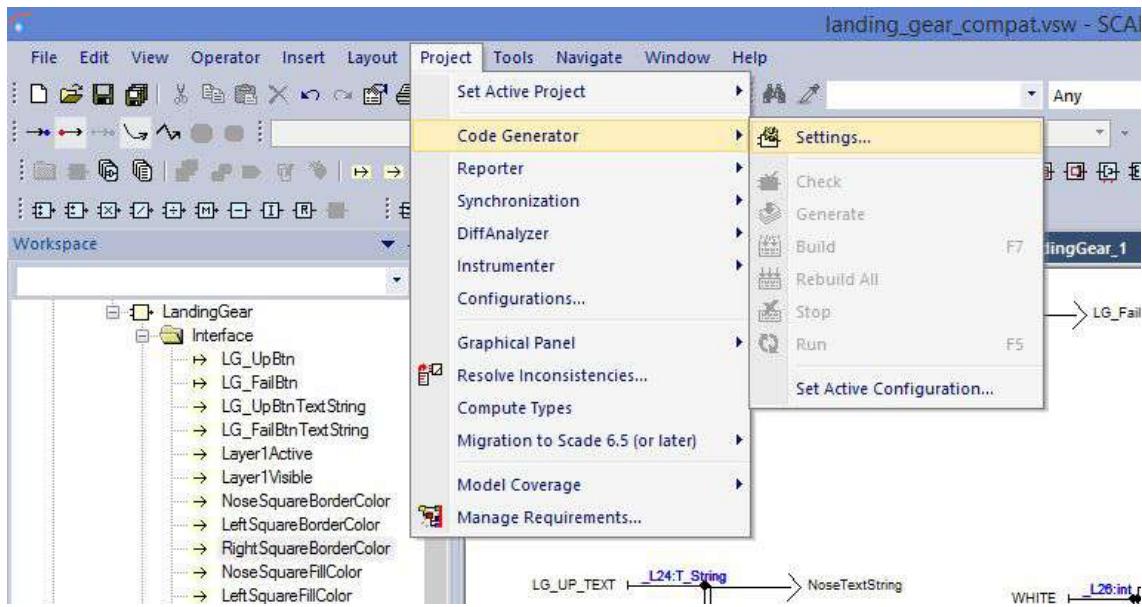
Obs: Os operadores lógicos utilizados foram **If, then else e followed by**. O “true” e o “false” utilizados são criados, usando-se a ferramenta marcada em vermelho:



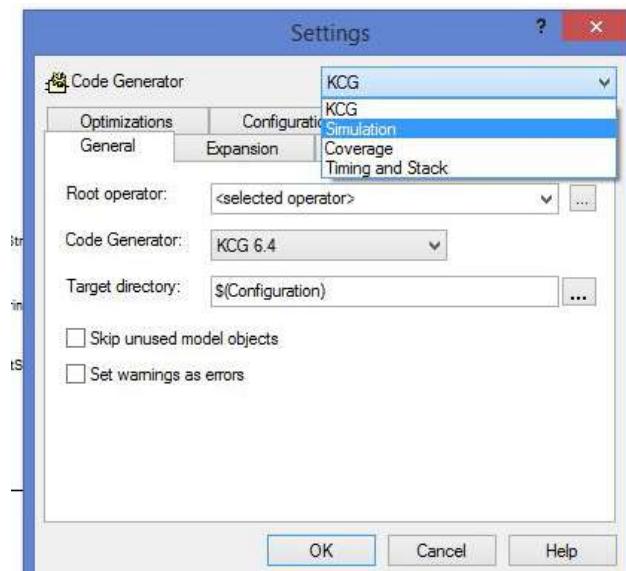
2.18 Agora, com as conexões feitas, clique duas vezes no operador *LandingGear* e insira o código abaixo:



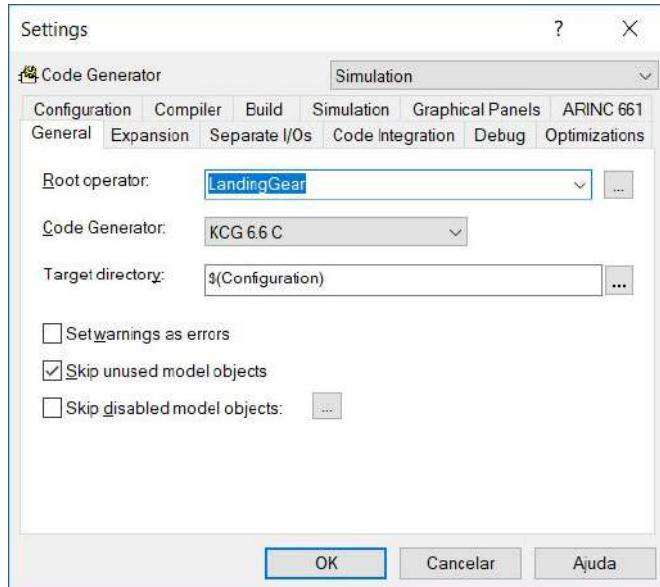
2.19 Agora, configure o “code generator”, através do menu Project/CodeGenerator/Settings:



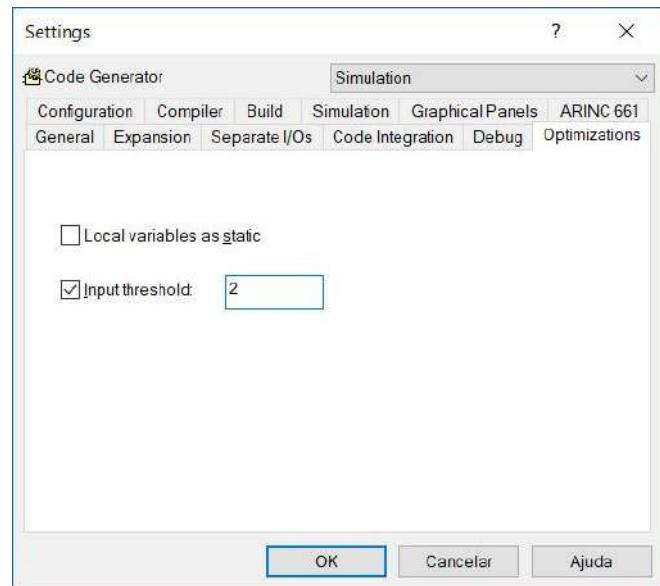
2.20 Agora, escolha a opção *Simulation* no dropdown menu:



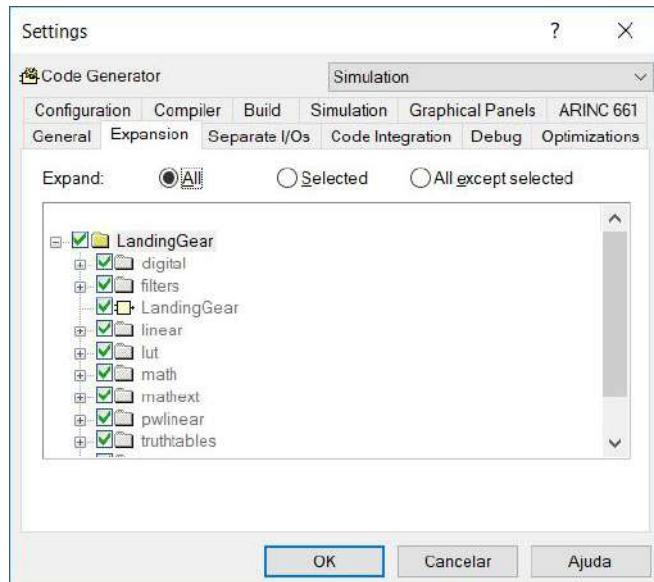
2.21 Na aba general, deixe todas as opções iguais a da imagem a seguir:



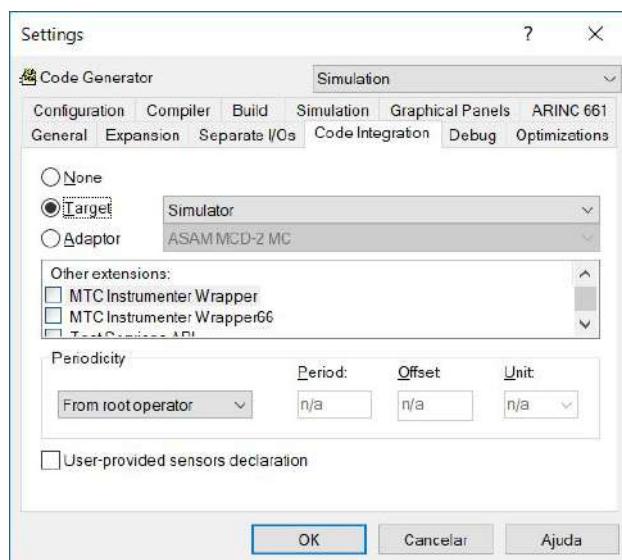
2.22 Escolha a aba *Optimizations* e deixe todas as opções iguais a imagem a seguir:



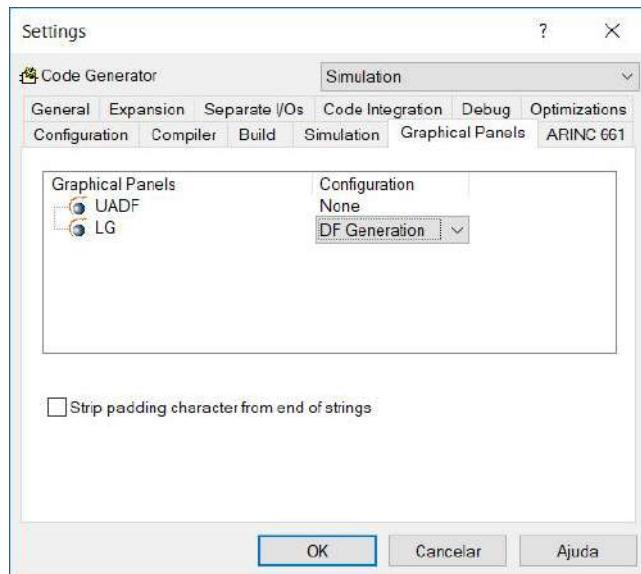
2.23 Faça o mesmo com a aba *expansion*:



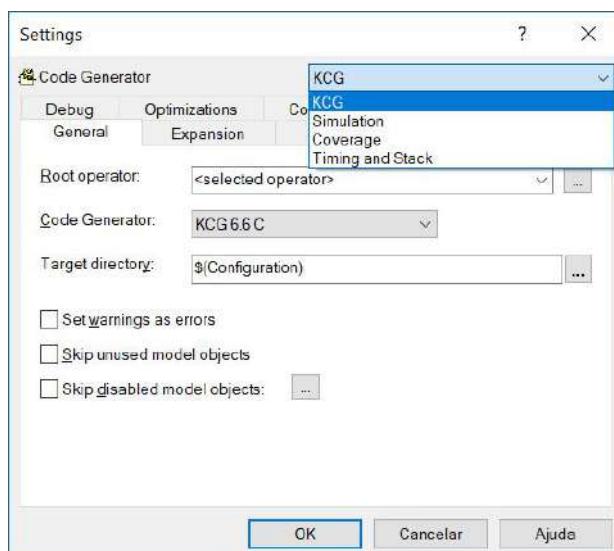
2.24 Faça o mesmo com a aba *code integration*:



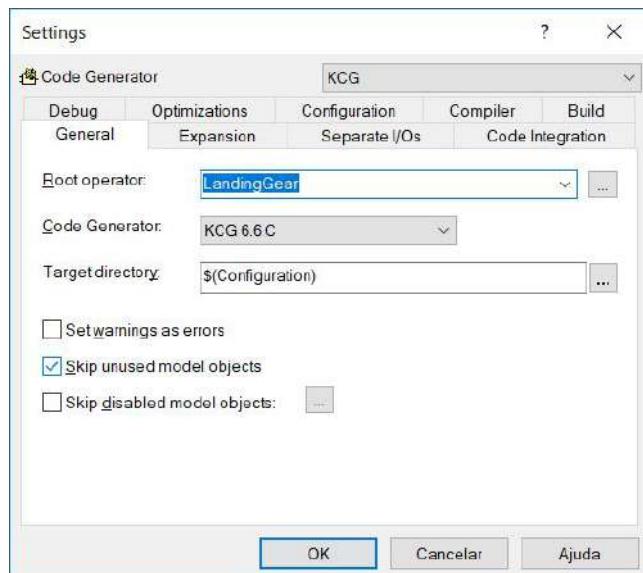
2.25 Faça o mesmo com a aba *Graphic Panels*:



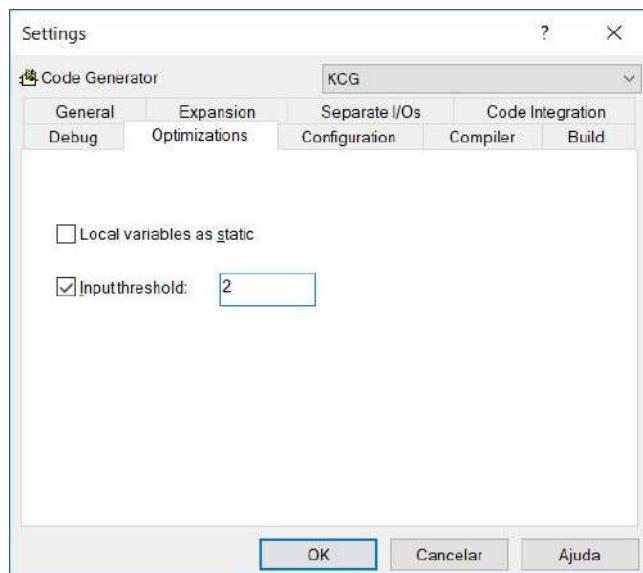
2.26 Agora mude a opção para KCG:



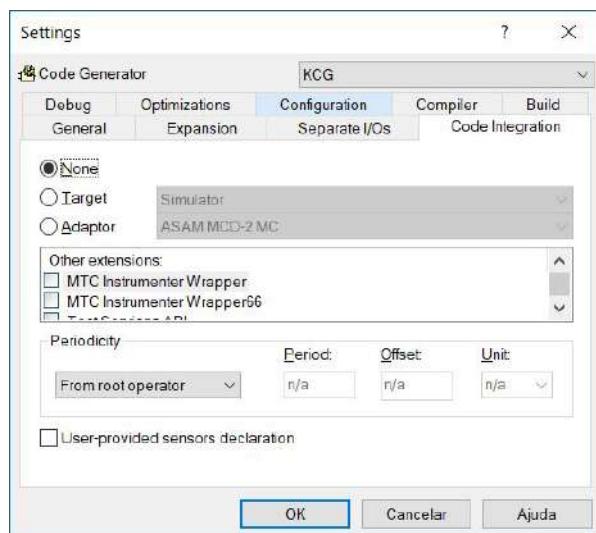
2.27 Deixe tudo igual a imagem abaixo na aba *General*



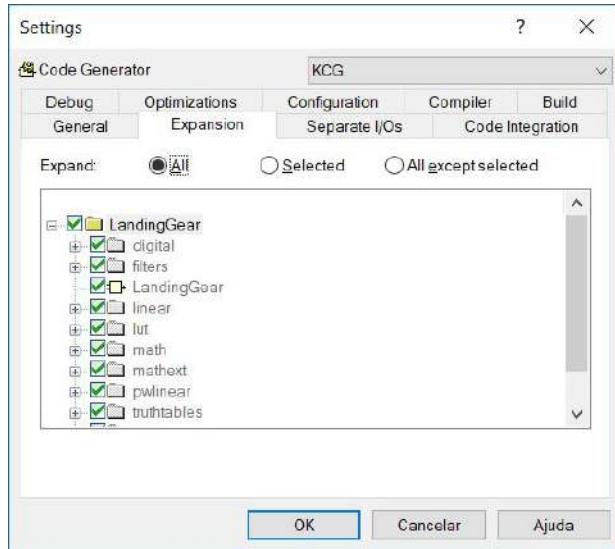
2.28 Na aba *optimization*, faça o mesmo:



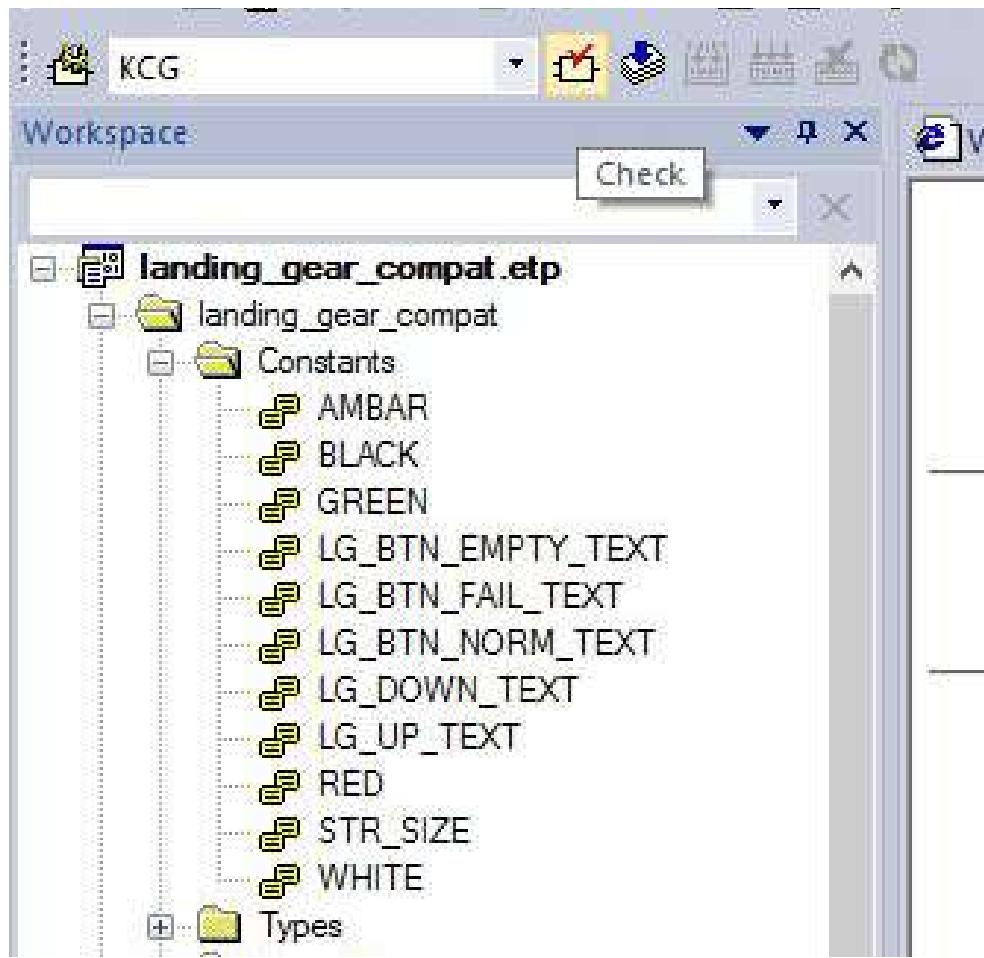
2.27 IDEM na aba *code integration*:



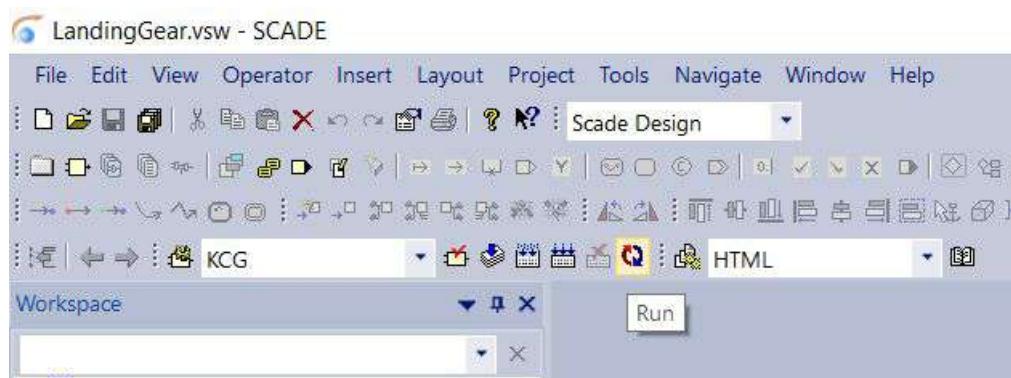
2.29 IDEM na aba *Expansion*:



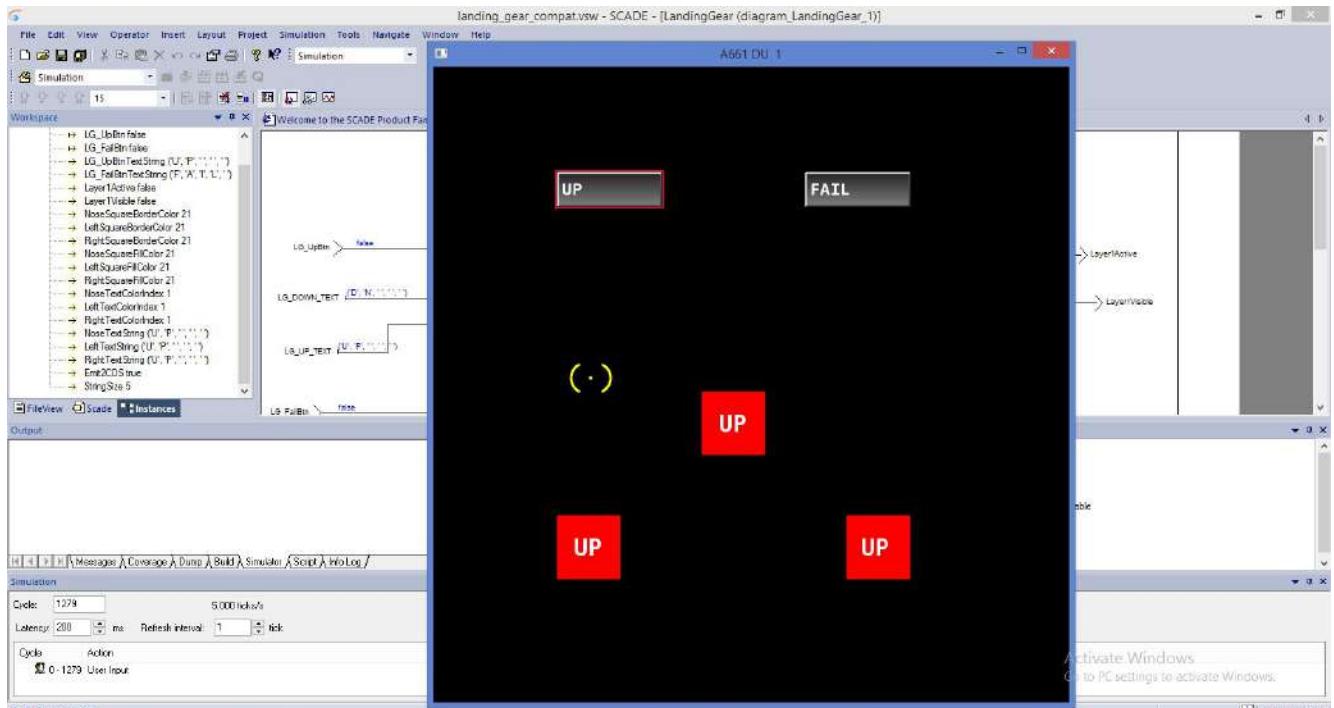
2.30 Ao final de tudo, clique em ok e vá no menu view/toolbars e marque a opção *code generator*. Agora, clique no botão *check*, para ver se existem erros:

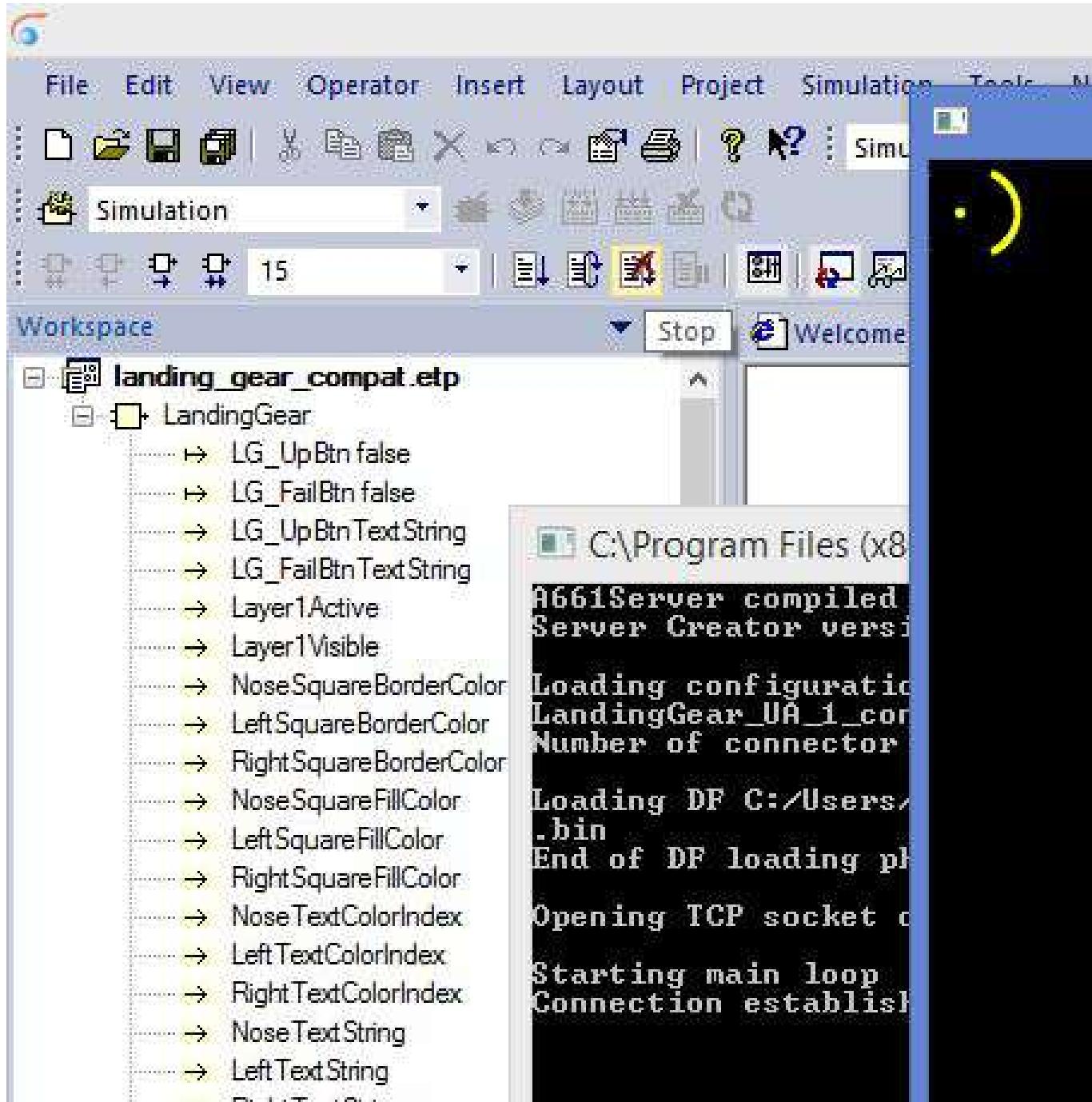


2.31 Agora, teste o programa, apertando o botão “run” conforme a figura a seguir:



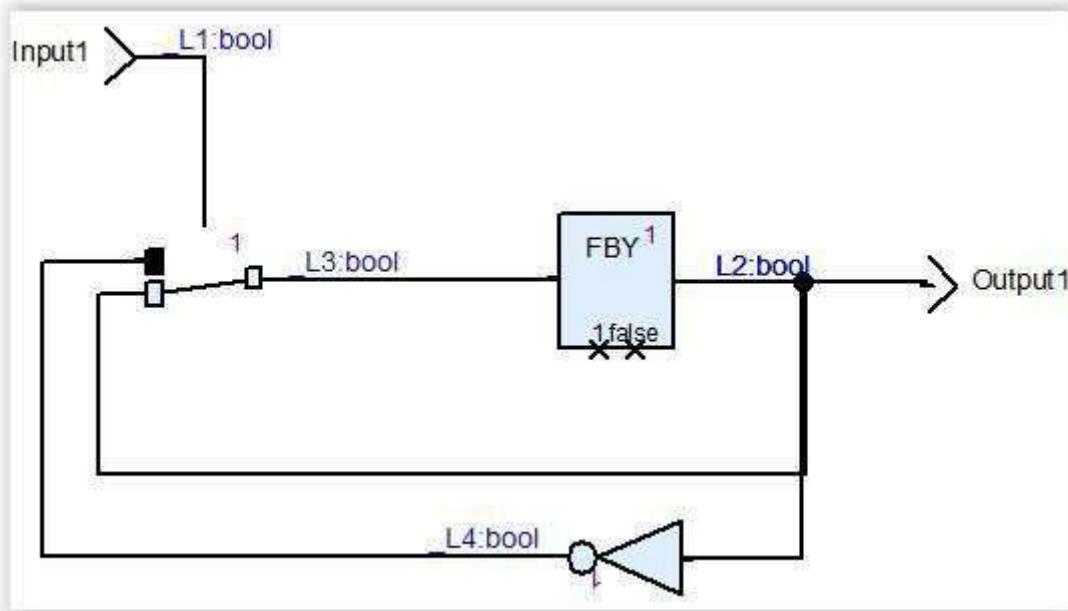
2.32 Após um tempo, um *prompt* e uma tela preta irão aparecer no monitor, clique no botão “GO” (contorno preto), **após aparecer a tela clique no menu Simulation e selecione a opção GO**, e a interface gráfica irá aparecer na tela preta:





Para parar a simulação, clique no botão “stop” (contorno vermelho na imagem acima) circulado em vermelho, ou clique no menu **Simulation** e selecione a opção **STOP**.

2.33 Agora, crie um novo operador chamado *Switch* e entre no seu

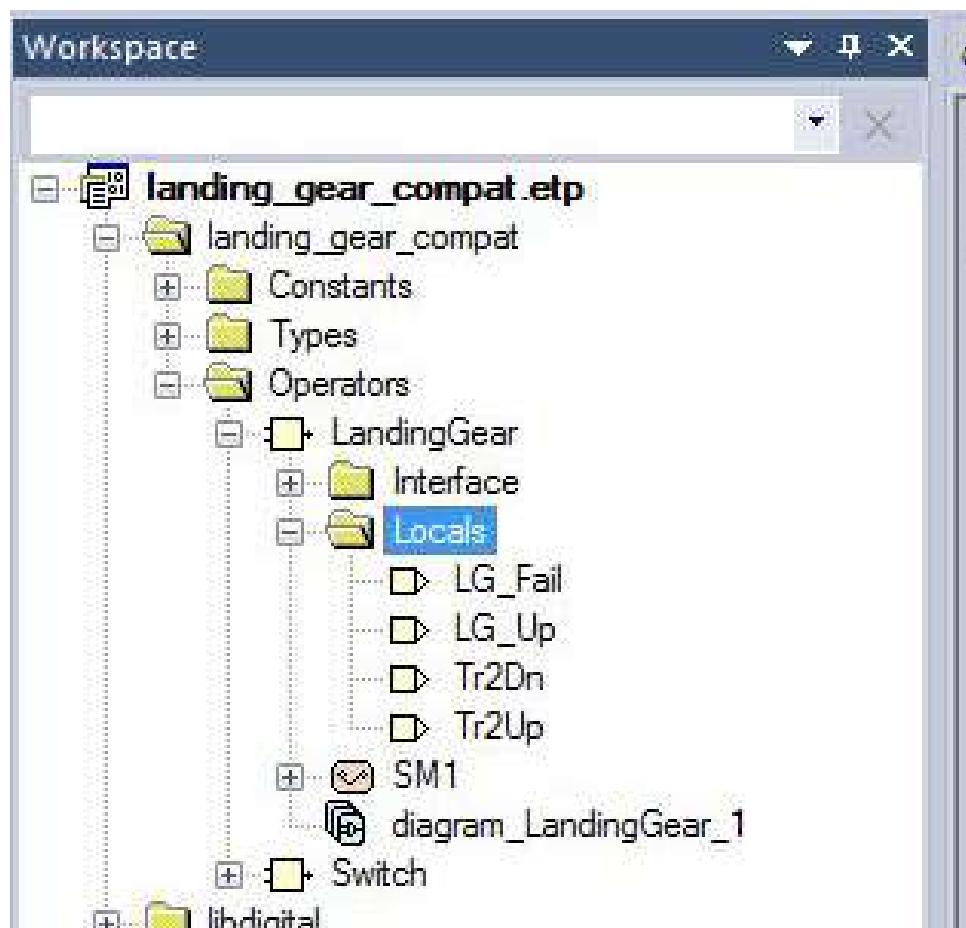


workspace. Agora, crie 1 *input* chamado *input1* e um *output* chamado *output1* neste operador. Depois, crie este layout (esquemático) dentro do mesmo:

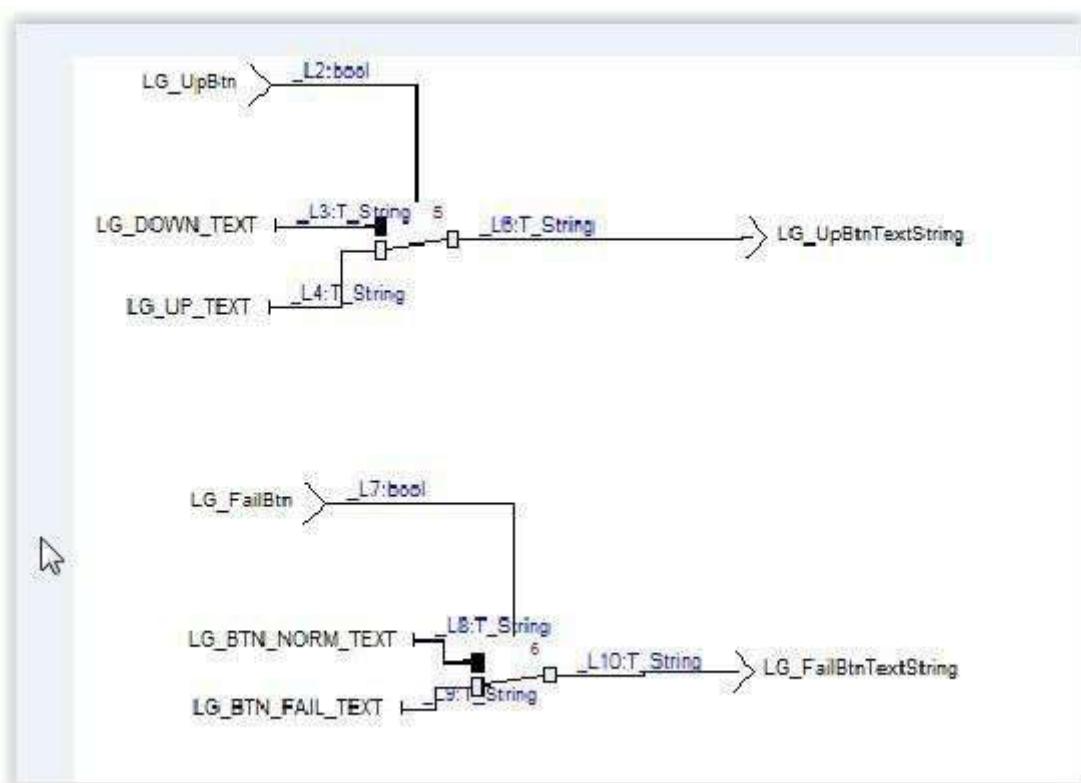
Este layout é responsável por fazer um input momentâneo ficar continuo

Ex: Um botão é apertado, quando isto acontece, o sinal que vem deste botão deixa de ser false,false,false.... e fica true,false,false..., quando este sinal entra no *switch*, o mesmo transforma o sinal de saída em true,true,true,true...

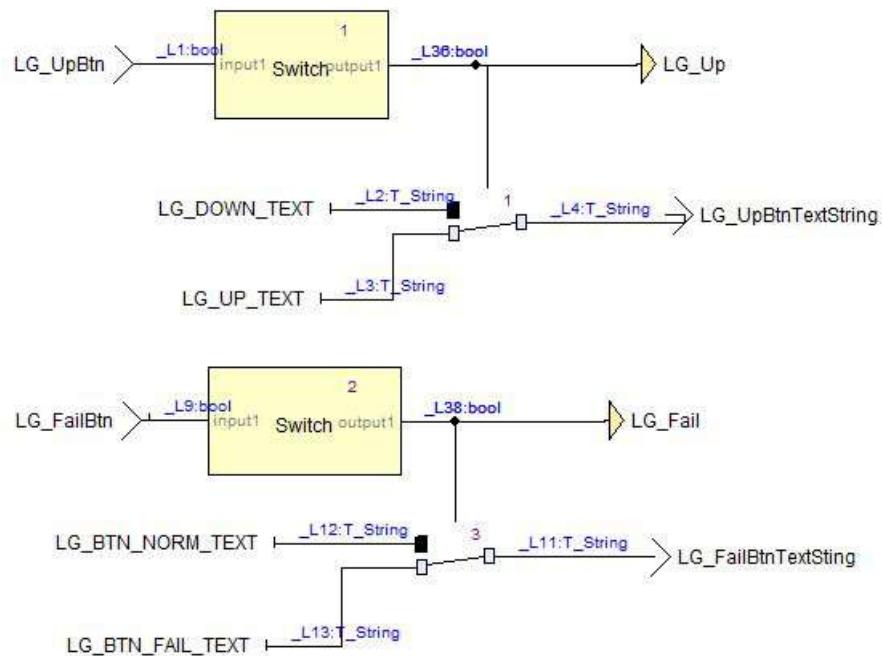
2.34 Agora, volte para o *workspace* do Operador *LandingGear* e crie duas variáveis locais do tipo *bool* chamadas *LG_Up* e *LG_Fail* (Do tipo *Out*).



Em seguida, mude o *layout* desta parte:



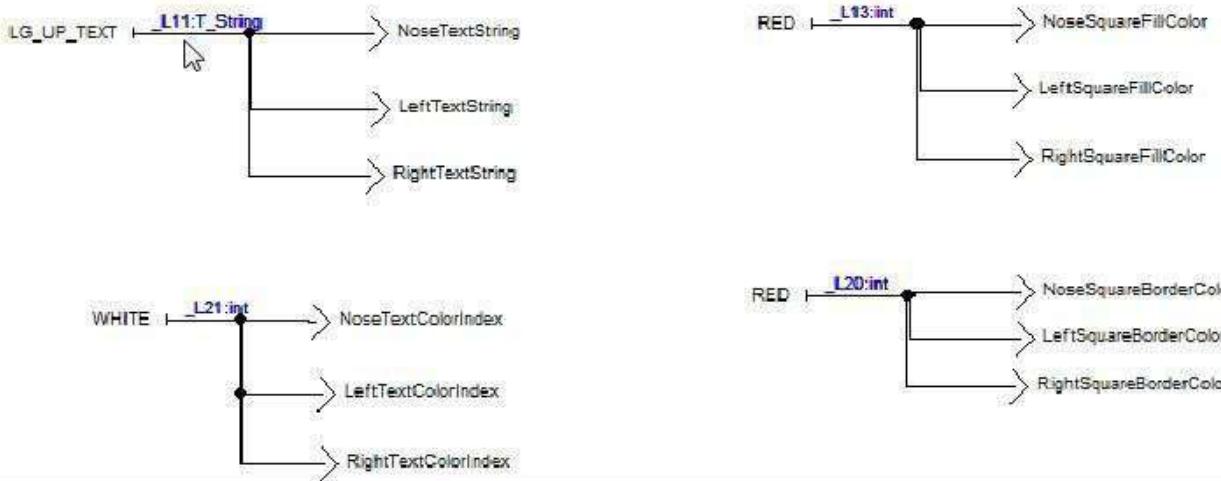
2.35 Para que ele fique assim (Arraste o operador *switch* para o *workspace*):



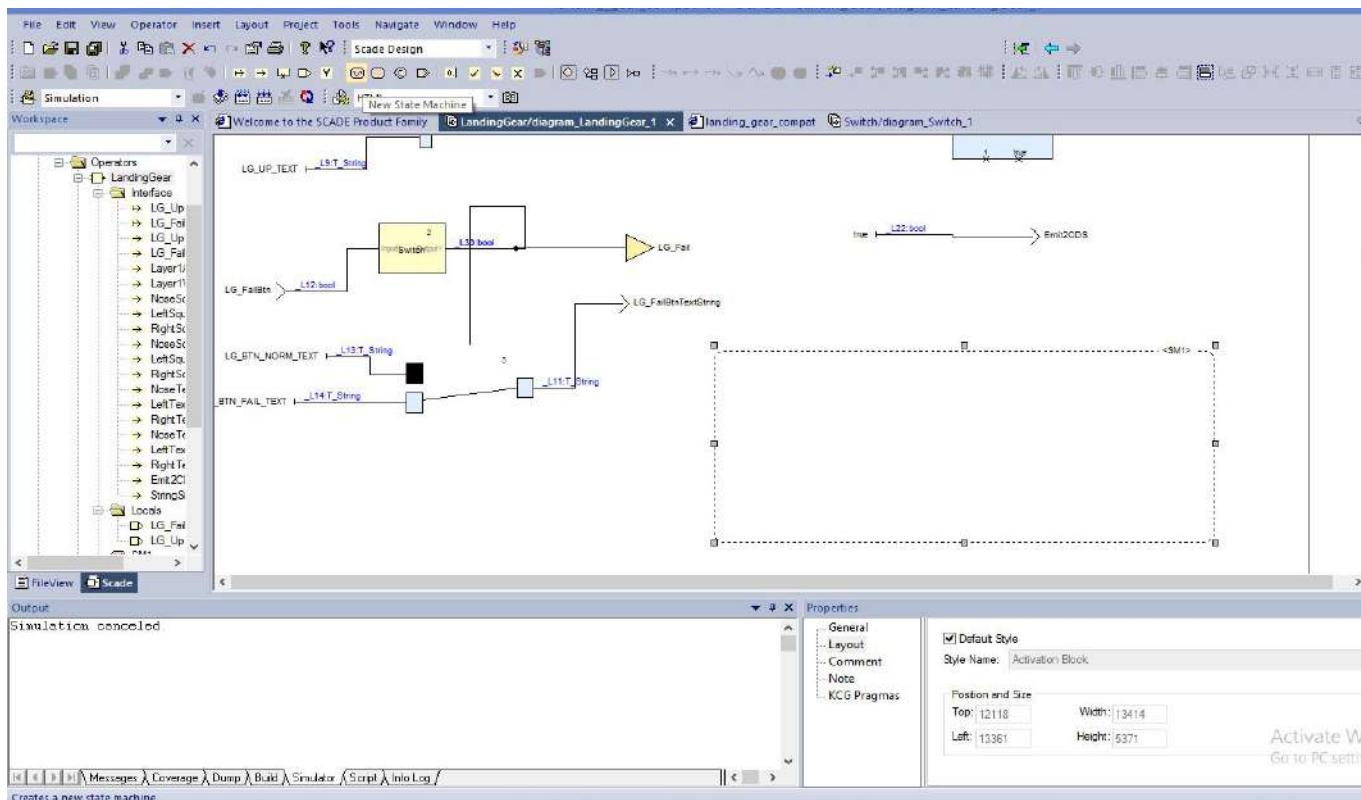
Desta maneira, quando o botão para descer ou subir o trem de pouso for acionado, a mudança será permanente e não momentânea.

Rode o programa, para testar o botão.

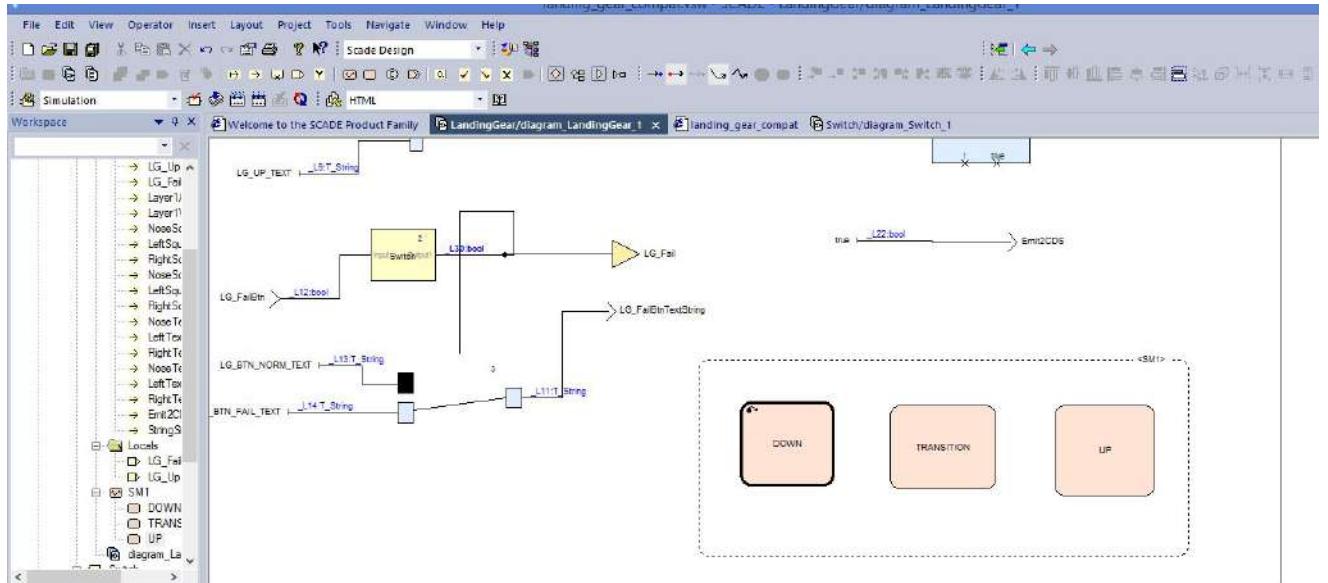
2.36 Agora, delete esta parte do *layout* do operador *LandingGear*:



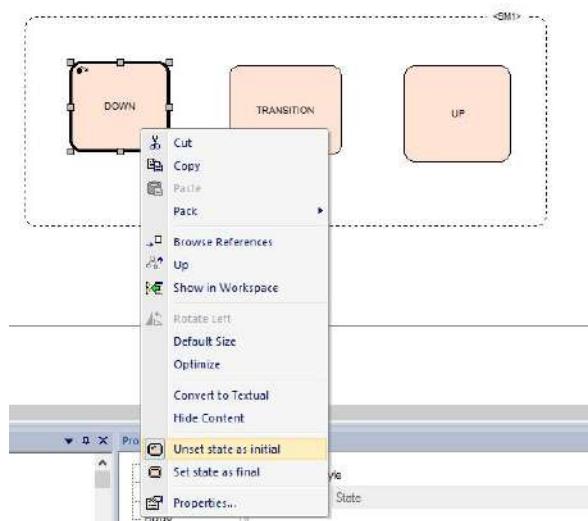
2.37 Agora, adicione uma Máquina de Estado ao *workspace* do operador *LandingGear*:



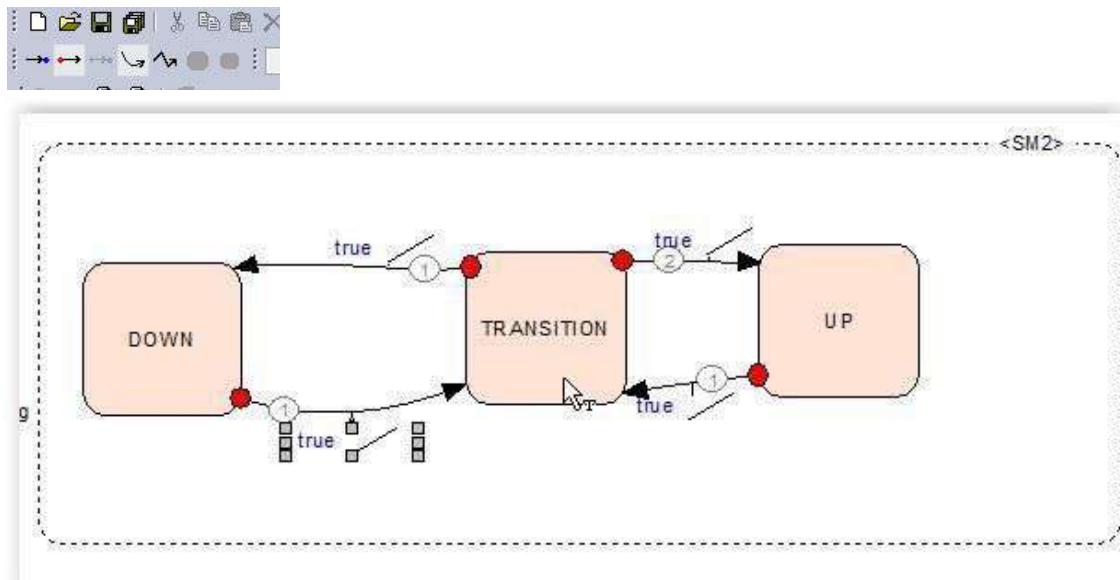
2.38 Adicione 3 estados dentro da Máquina de Estado com os seguintes nomes : DOWN,TRANSITION e UP e crie duas variáveis locais chamadas Tr2Dn e Tr2Up, do tipo bool:



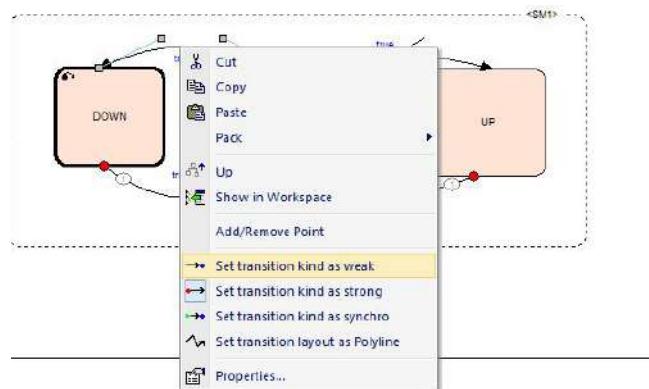
2.39 Clique com o botão direito no estado down e selecione a opção “Set State as initial”:



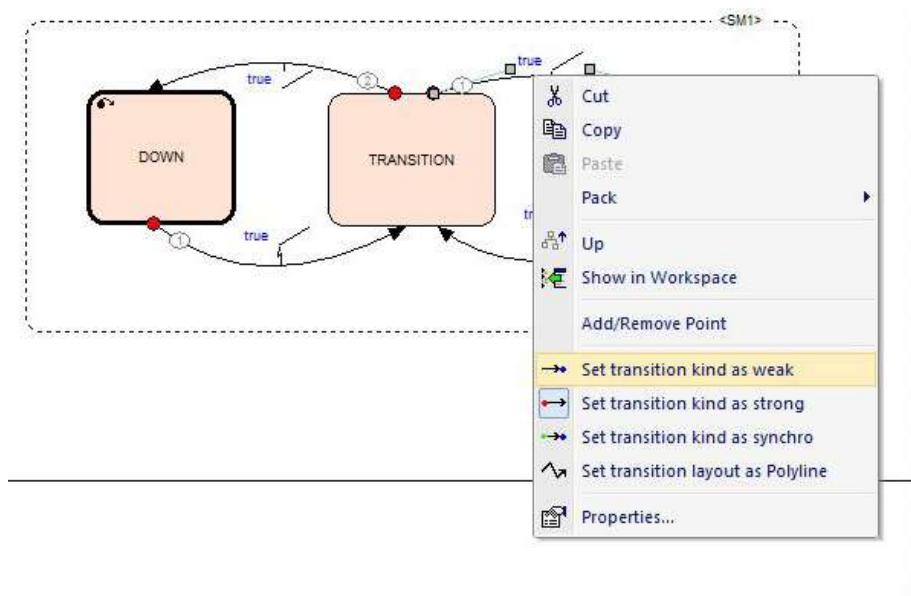
2.40 Agora, faça a conexão entre os estados, de acordo com a imagem a seguir (Utilize a barra “State Machine”):



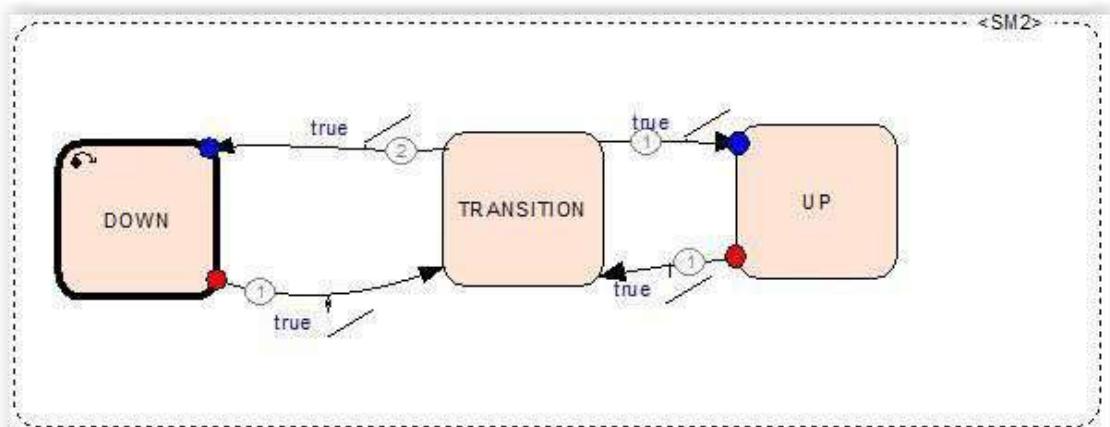
2.41 Agora, clique com o botão direito na conexão superior esquerda e selecione a opção “Set transition kind as weak”:



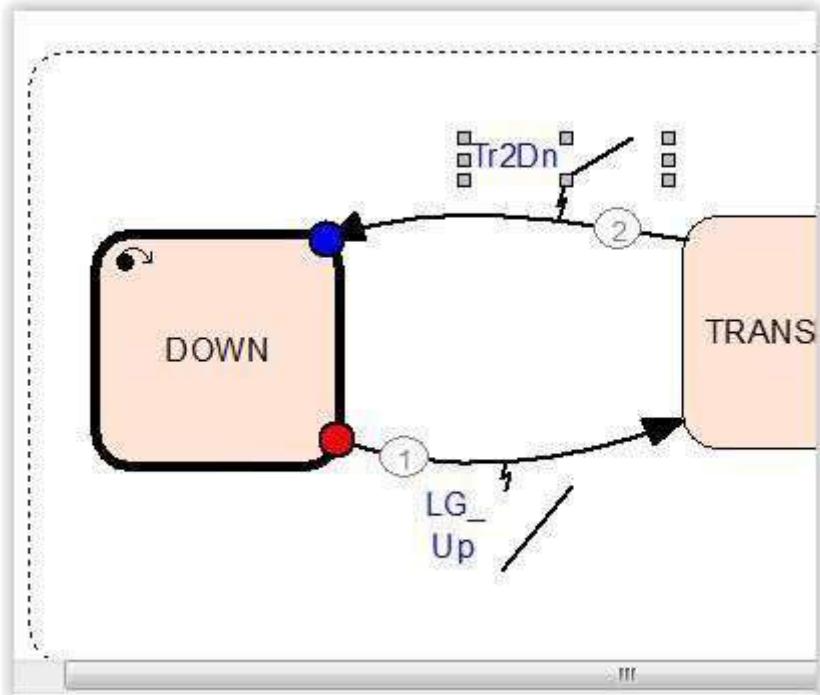
2.42 Faça o mesmo com a conexão direita superior



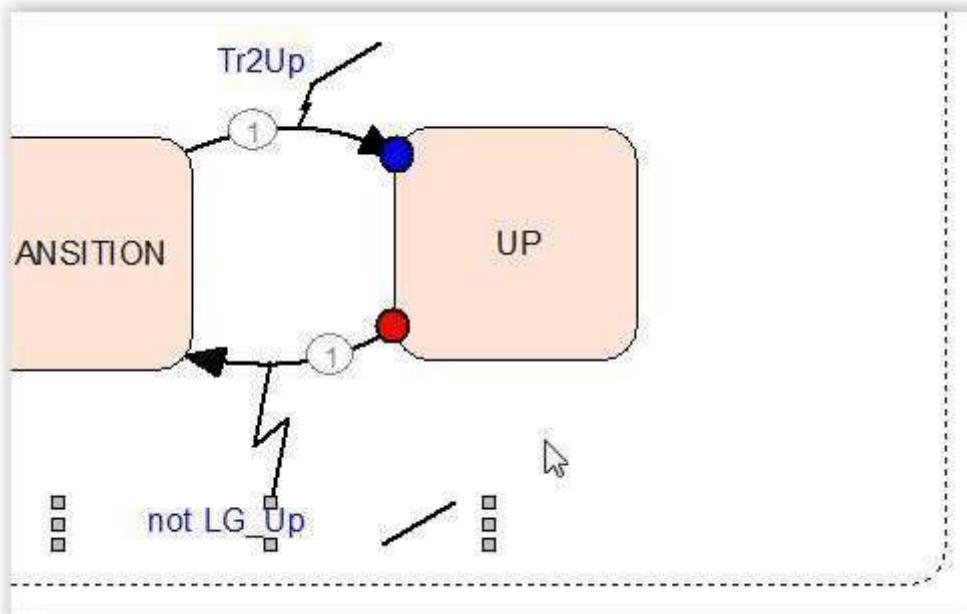
2.43 Perceba que os círculos superiores mudaram de posição e de cor



2.44 Agora, mude o valor da esquerda superior de true para Tr2Dn e o valor da esquerda inferior para LG_Up:

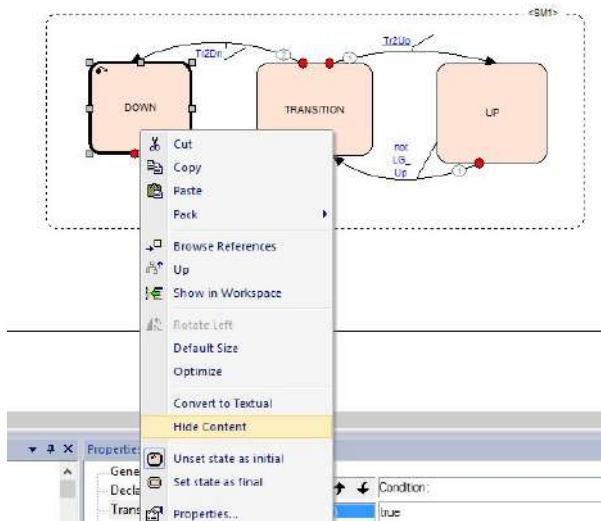


2.45 E o valor direito superior para Tr2Up e direto inferior para “not LG_Up”:

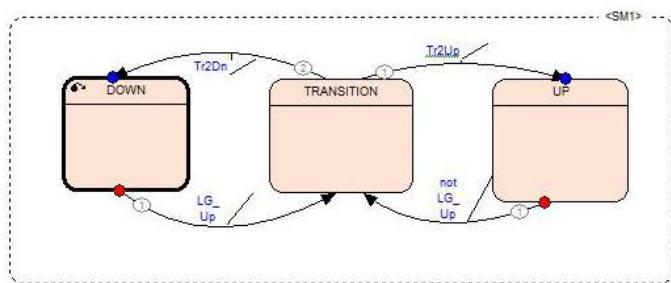


Obs: O *not* antes do LG_UP é um operador lógico utilizado para inverter o valor de true para false e vice-versa.

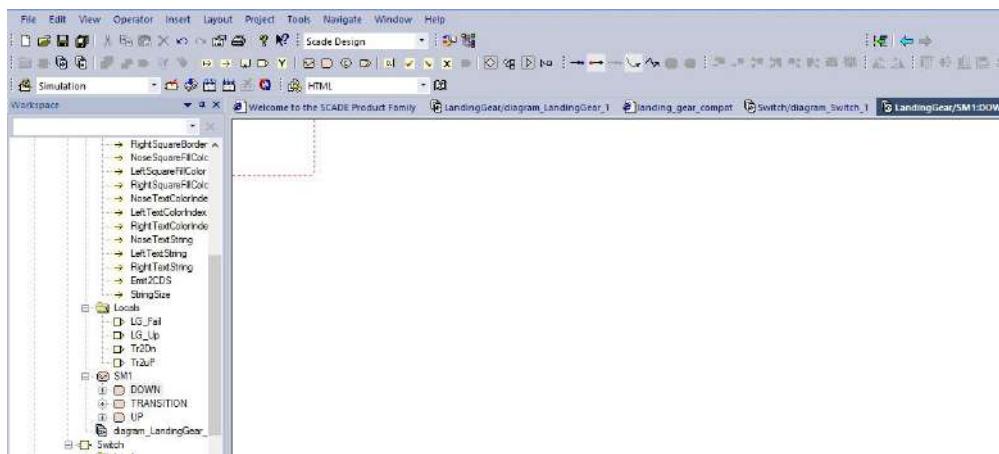
2.46 Agora, clique com botão direito em cada um dos estados e escolha a opção “*Hide Content*”.



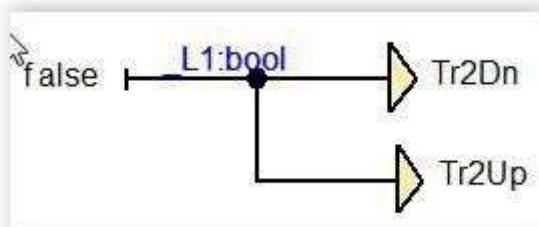
Para que eles fiquem com esta aparência:



2.47 Agora, clique duas vezes no estado *down*, e perceba que o workspace deste estado se abrirá.

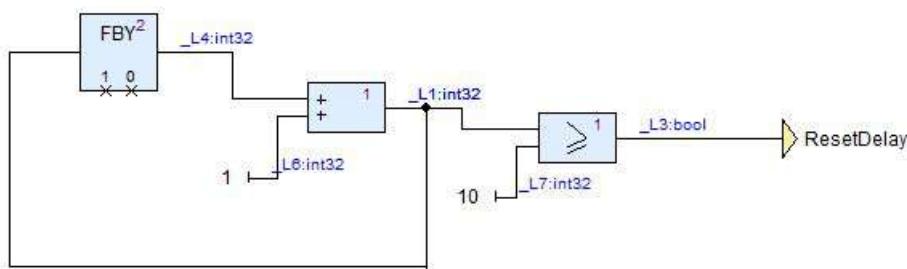


2.48 Agora, adicione este código dentro do workspace do estado *down*:



Adicione o mesmo código no estado UP !!!

2.49 Agora, clique duas vezes no estado *TRANSITION*, crie uma variável local deste estado chamada “*ResetDelay*” e adicione o seguinte código:

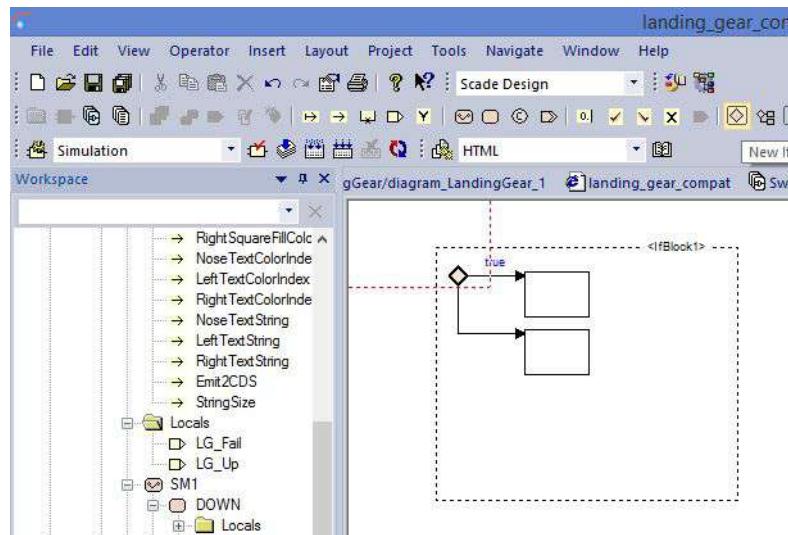


Este código funciona como um contador para impedir que ocorra uma mudança de estado antes desta mudança ter ocorrido por completo.

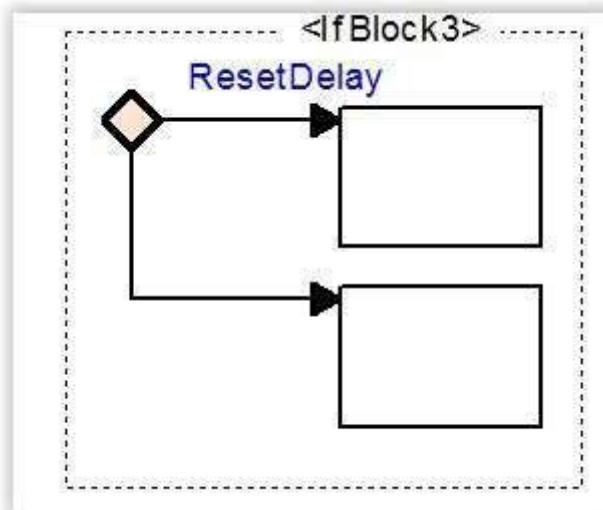
Obs: O número 1 e o número 10 foram criados, usando a ferramenta circulada em preto (*New Textual Expression*) e os blocos utilizados foram o *Followed by*, *Plus* e *Greater than or equal*.



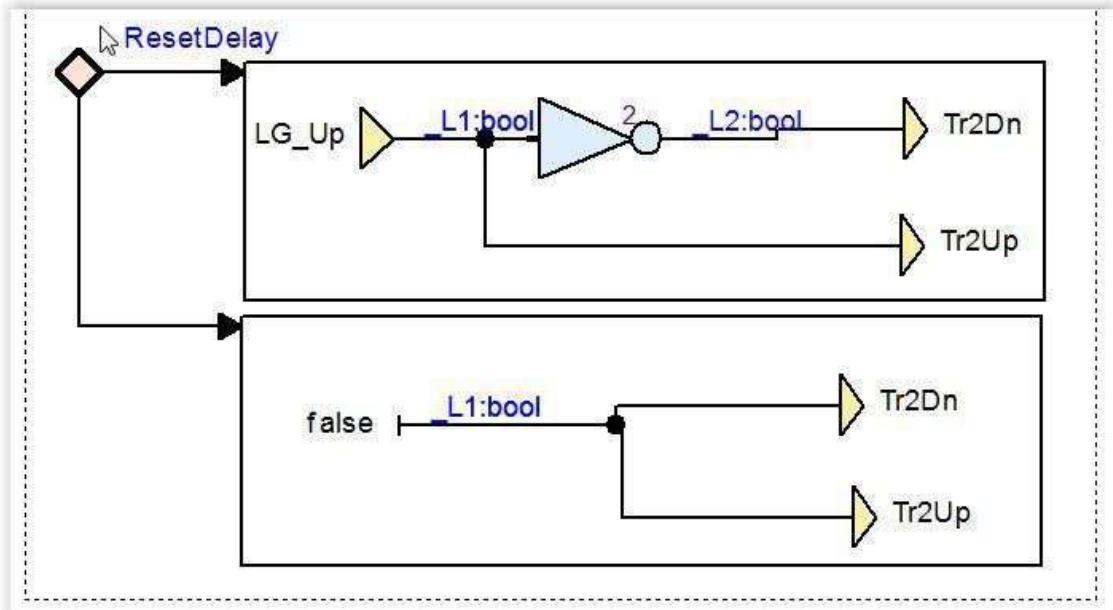
2.50 Agora, adicione um novo bloco IF dentro do estado *transition*



Mude o valor em azul para *ResetDelay*.

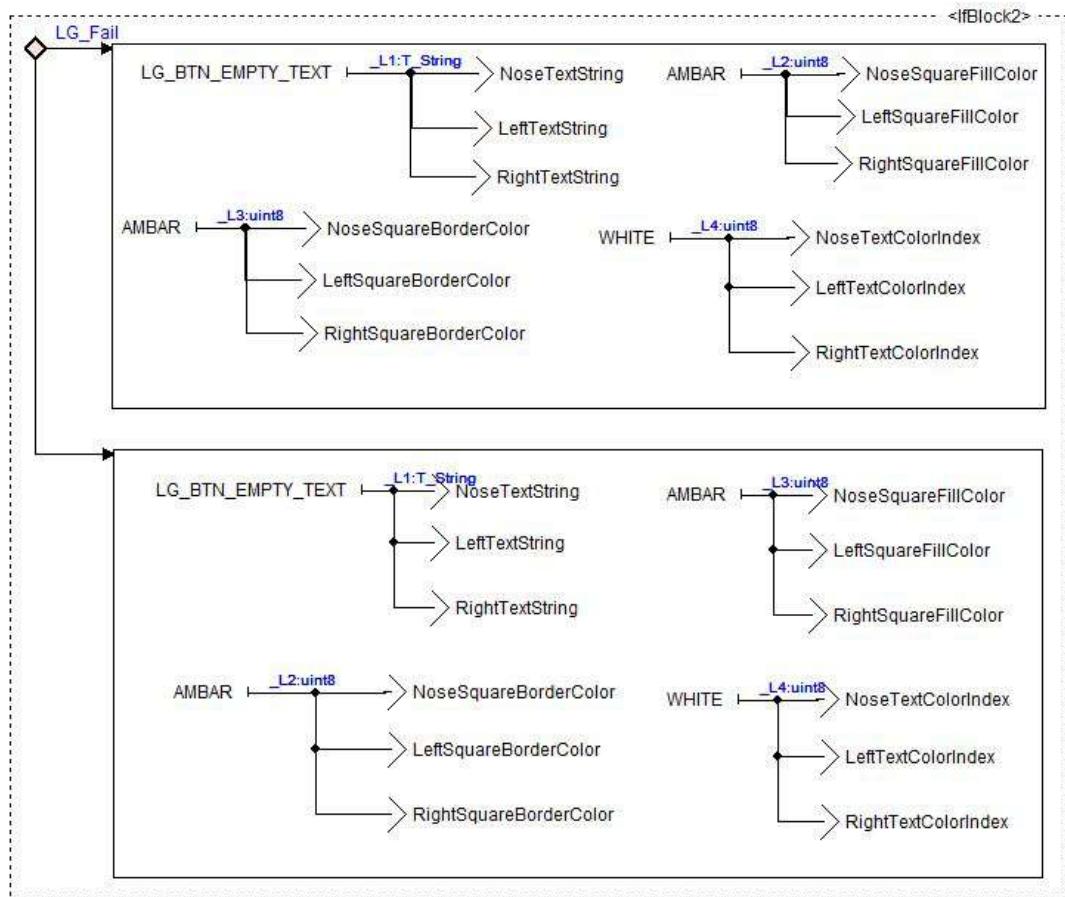


2.51 Agora, adicione este código dentro do bloco:



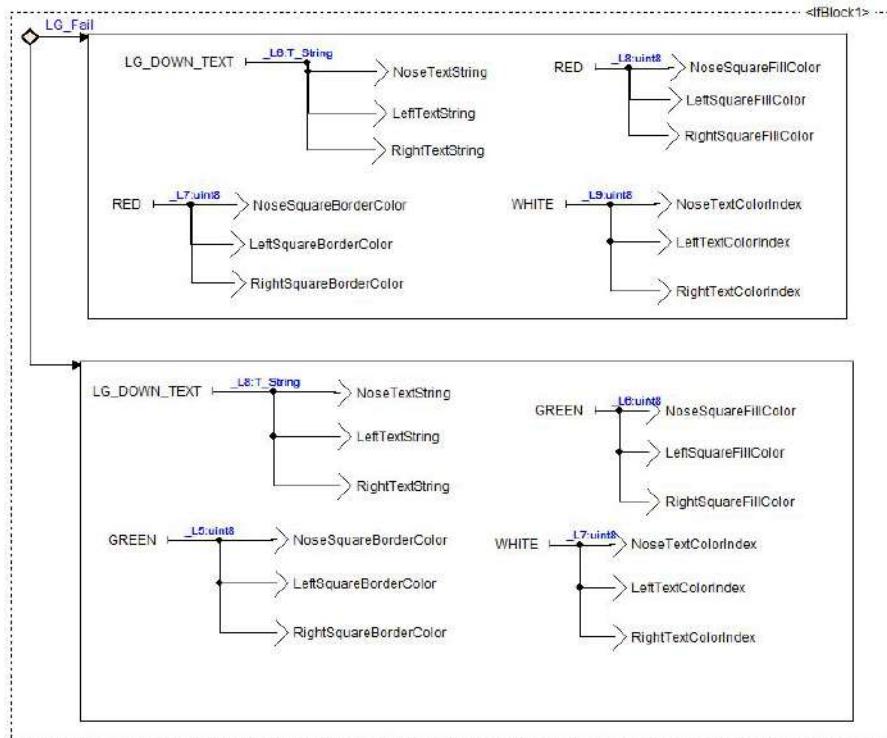
Este bloco será responsável por mudar o estado assim que o contador anteriormente elaborado acusar a mudança. Desta forma, redefine-se o valor de cada uma das variáveis mostradas para que reflitam a mudança.

2.52 Agora, adicione mais um bloco if dentro do estado *TRANSITION* e mude o valor em azul para *LG_Fail* e adicione o código da imagem a seguir:

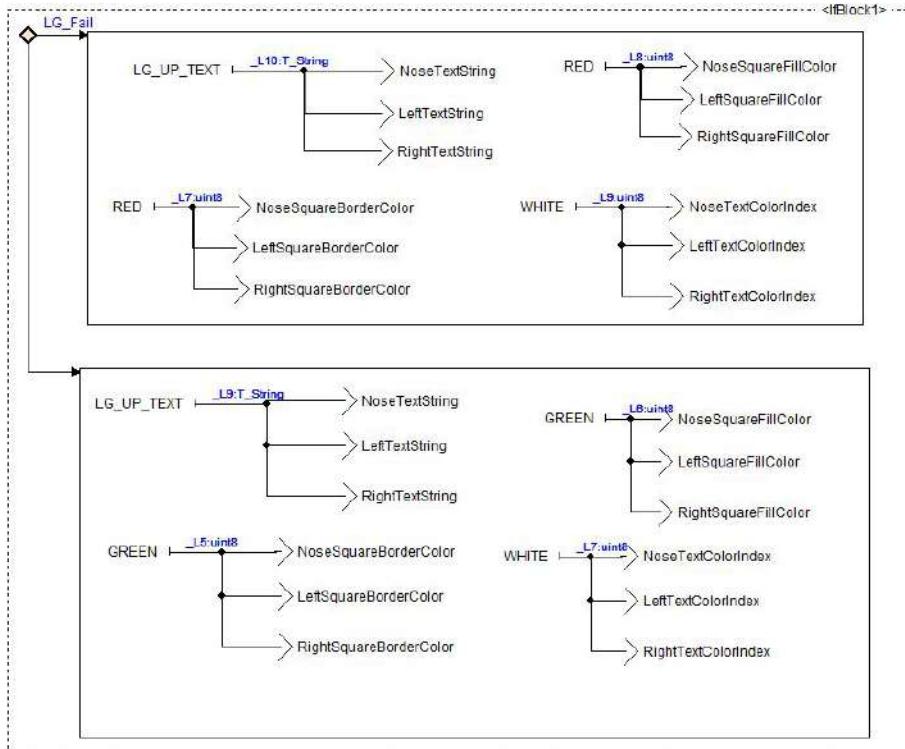


Este bloco será responsável pelas mudanças visuais durante a transição de um estado para o outro.

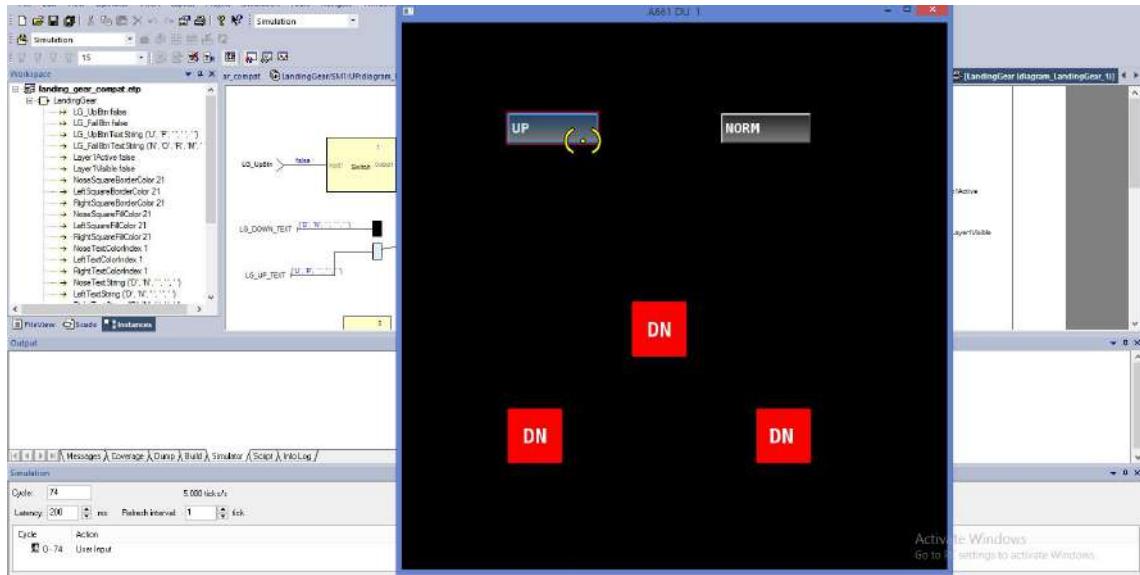
2.53 Agora, vá para o estado *Down*, adicione um novo bloco *if* e adicione o mesmo código, mudando apenas o *LG_EMPTY_TEXT* para *LG_DOWN_TEXT* e as cores



2.54 Faça o mesmo no estado *UP*, mas mude o *LG_DOWN_TEXT* para *LG_UP_TEXT* e as cores

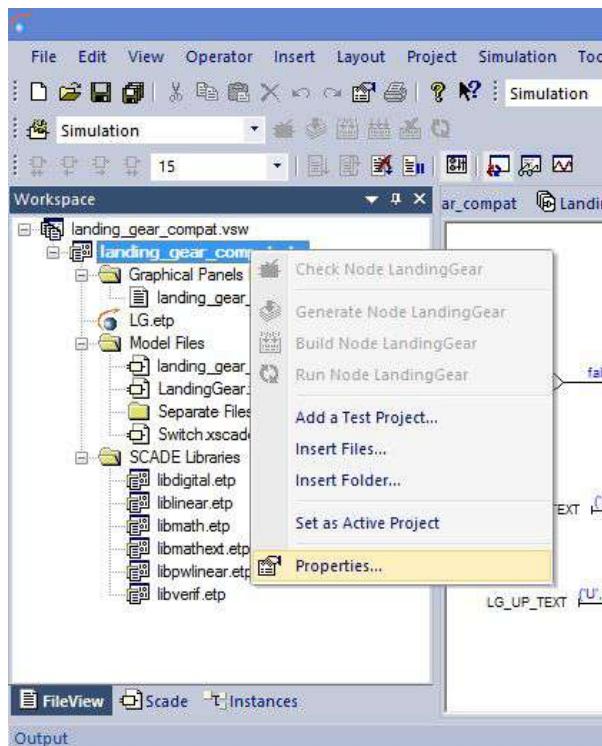


2.55 Agora, se tudo foi realizado corretamente, o programa poderá ser rodado sem nenhum erro. Ao final, ele deverá ficar com a seguinte aparência:

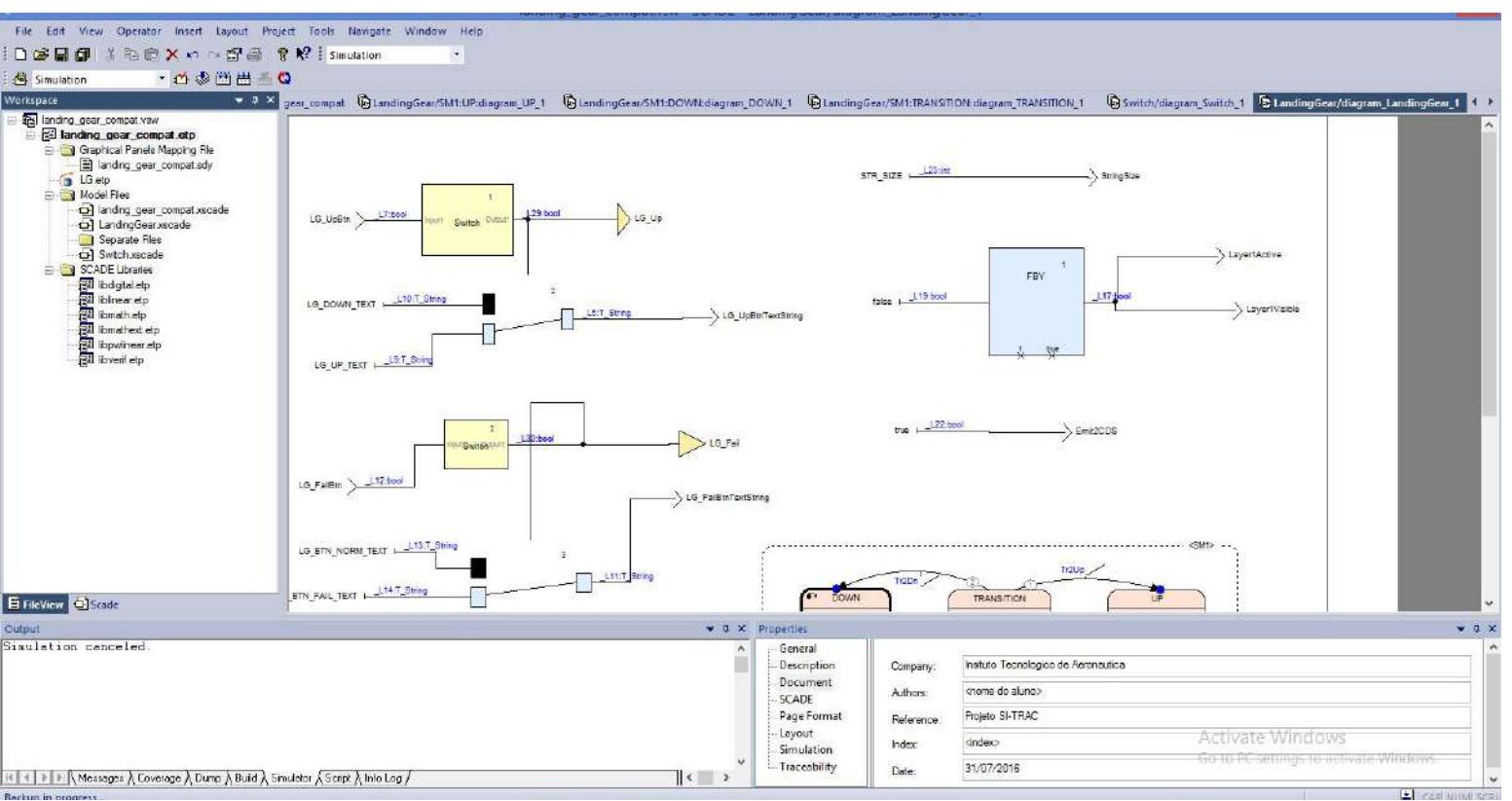


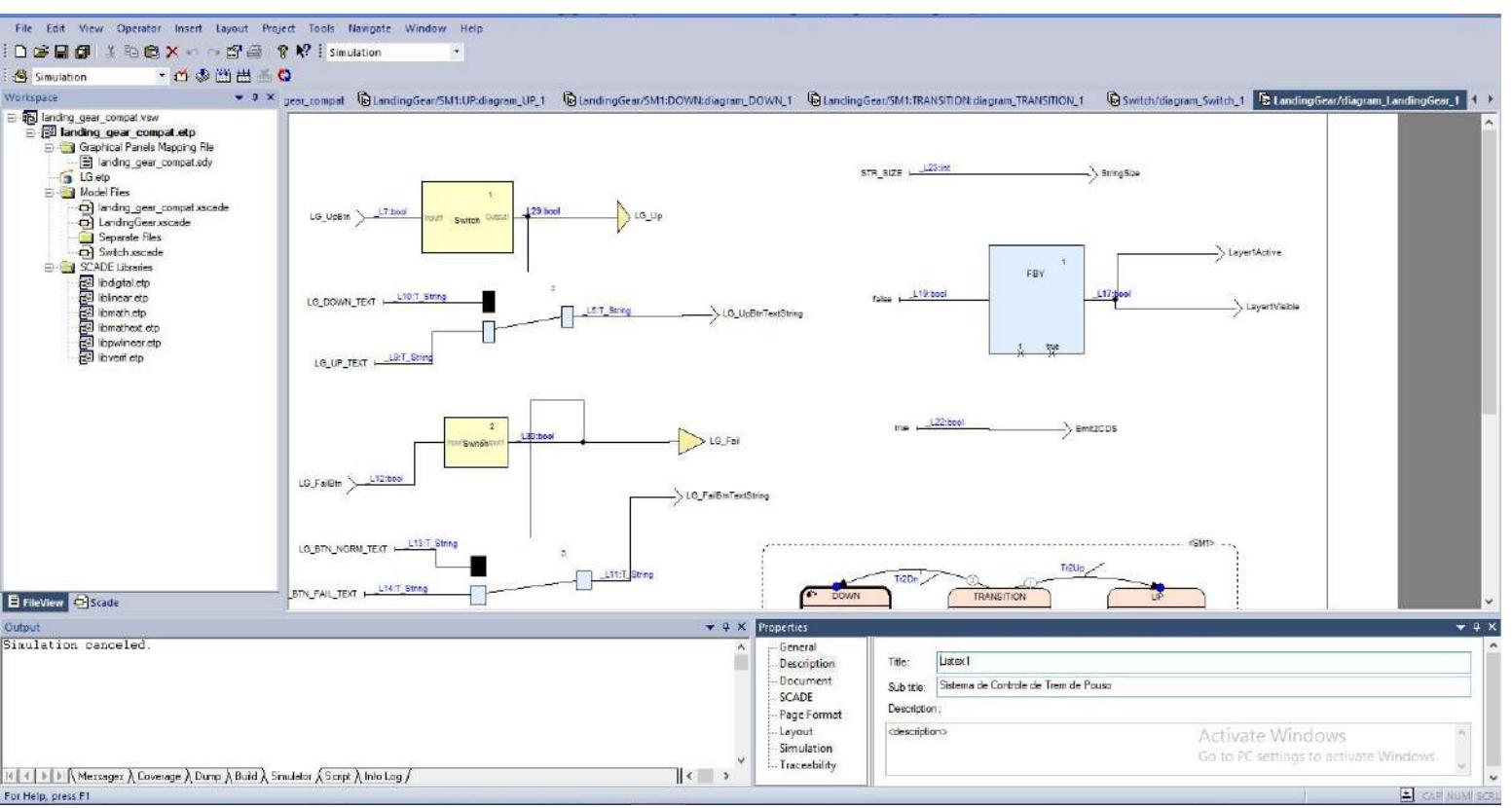
3 Agora, você deve criar o *Report* do *User Application* (UA). Para criá-lo, é necessário preencher certos dados.

Para preenchê-los, clique em FileView e, com o botão direito, no *LandingGear.etc* e vá em *properties*:

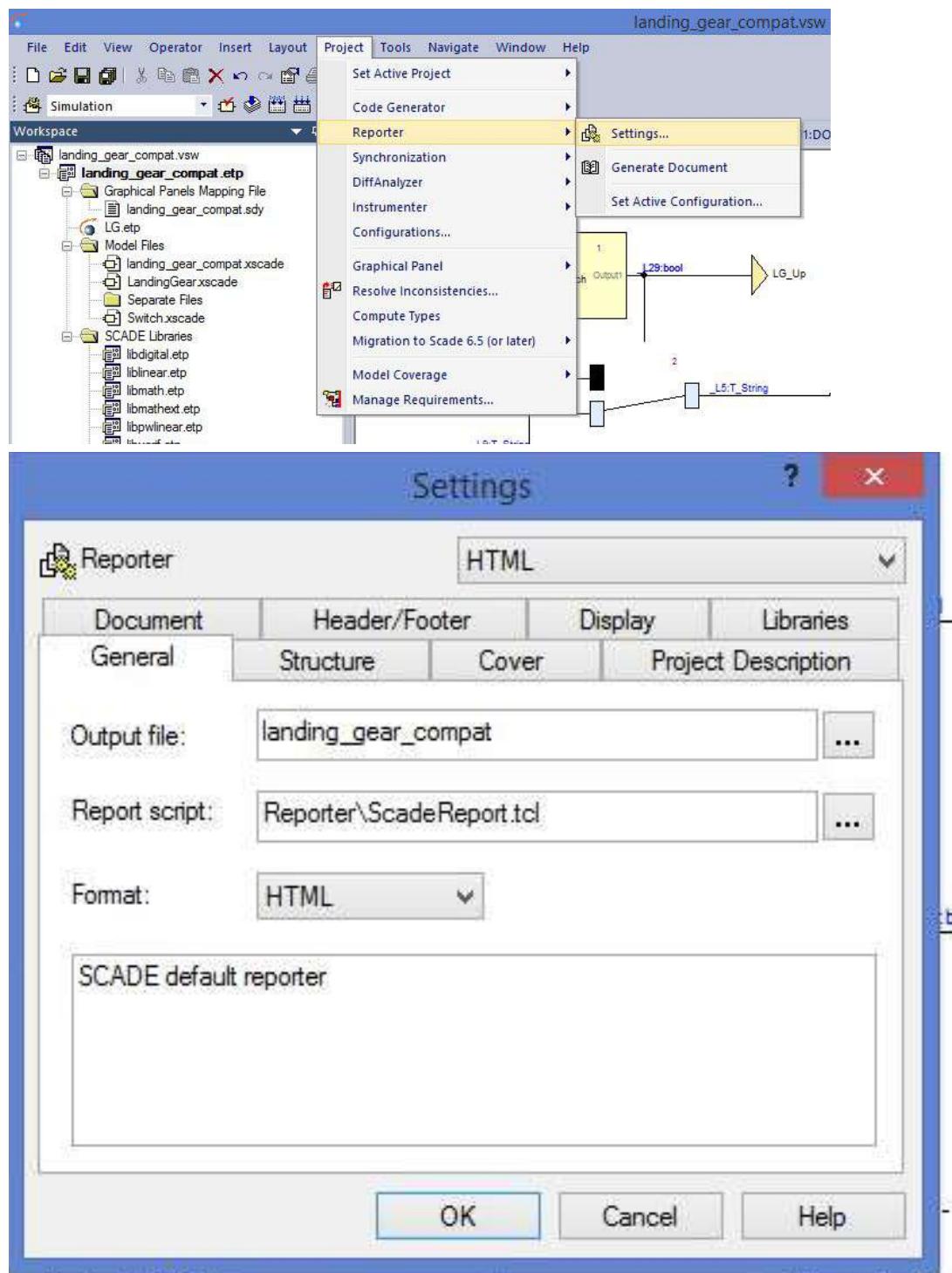


3.1 Preencha os campos das guias *Description* e *Document*

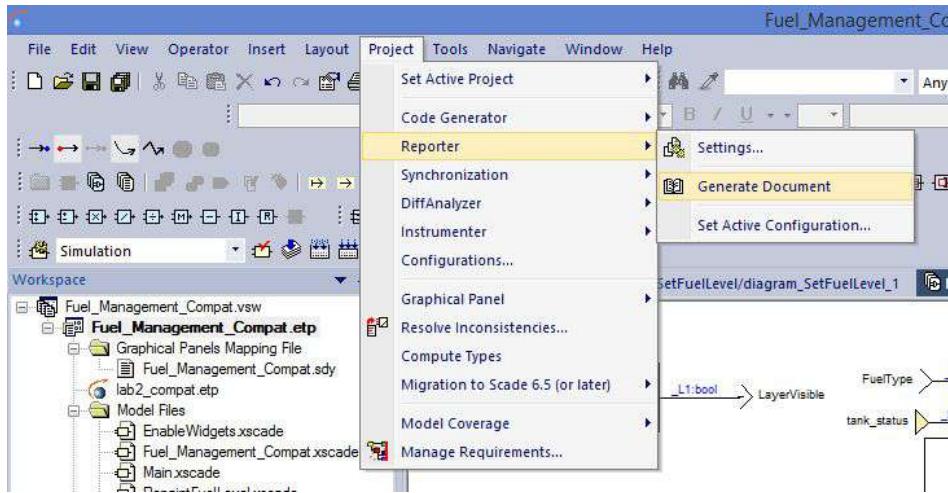




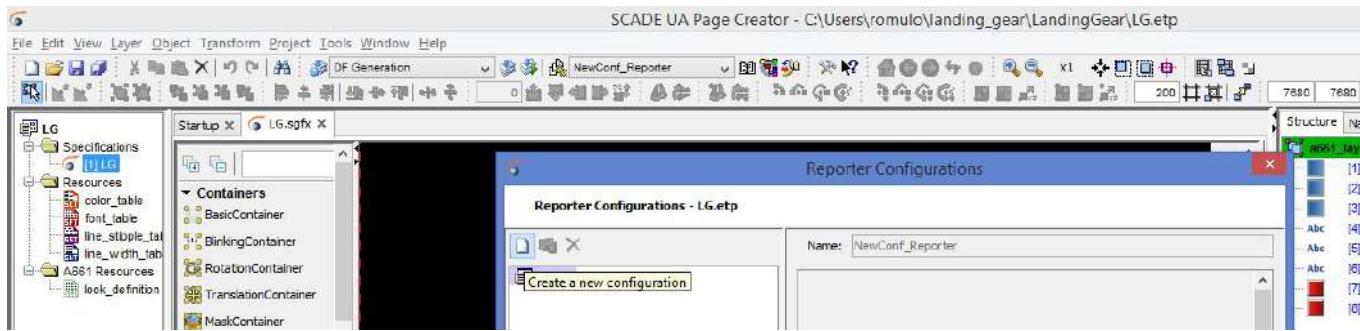
3.2 No menu *Project/Reporter/settings*, existem outras opções que podem ser modificadas, caso seja preciso:



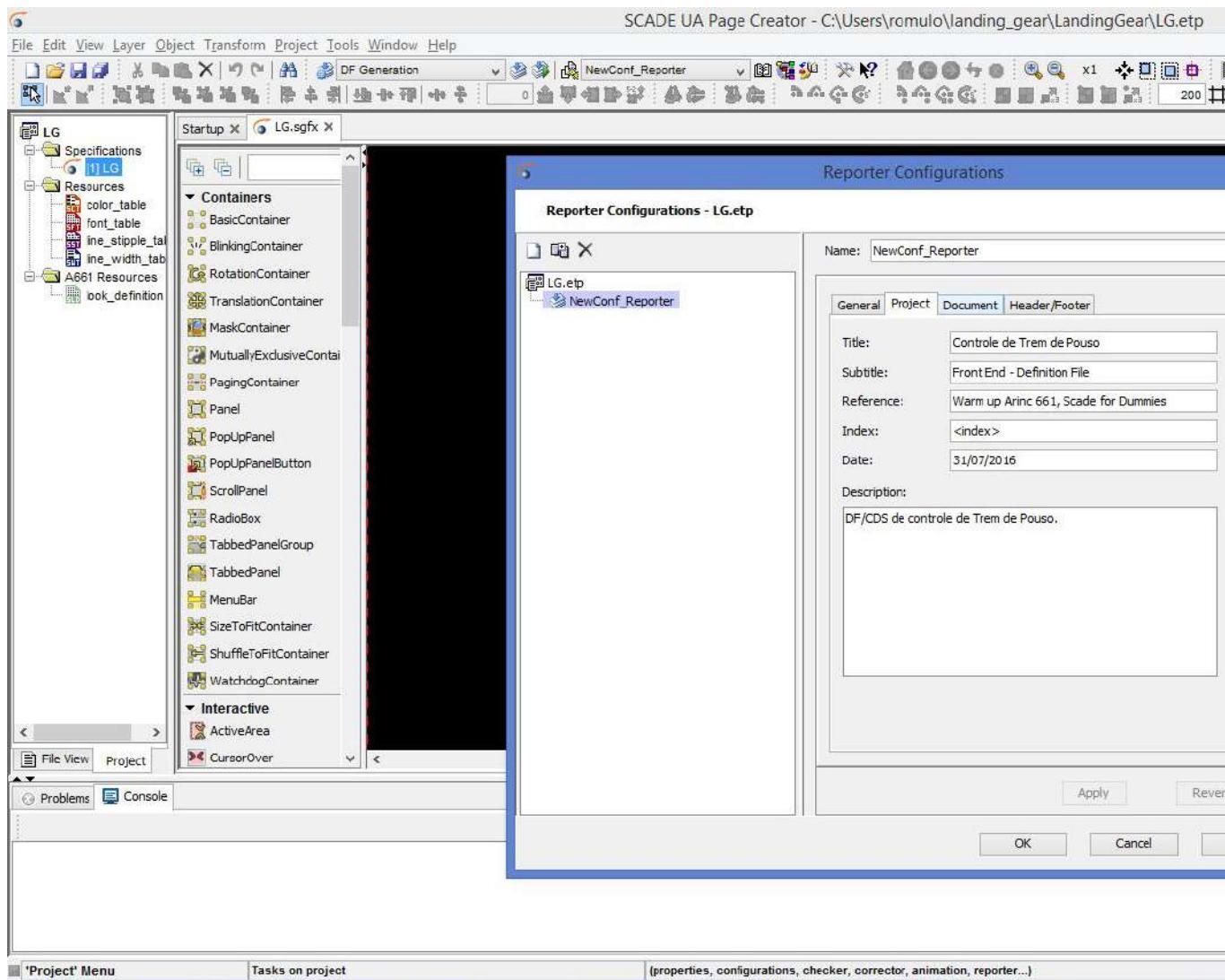
3.3 Clique em *Generate Document* e um *Report* será gerado que ficará dentro da pasta do seu Projeto:



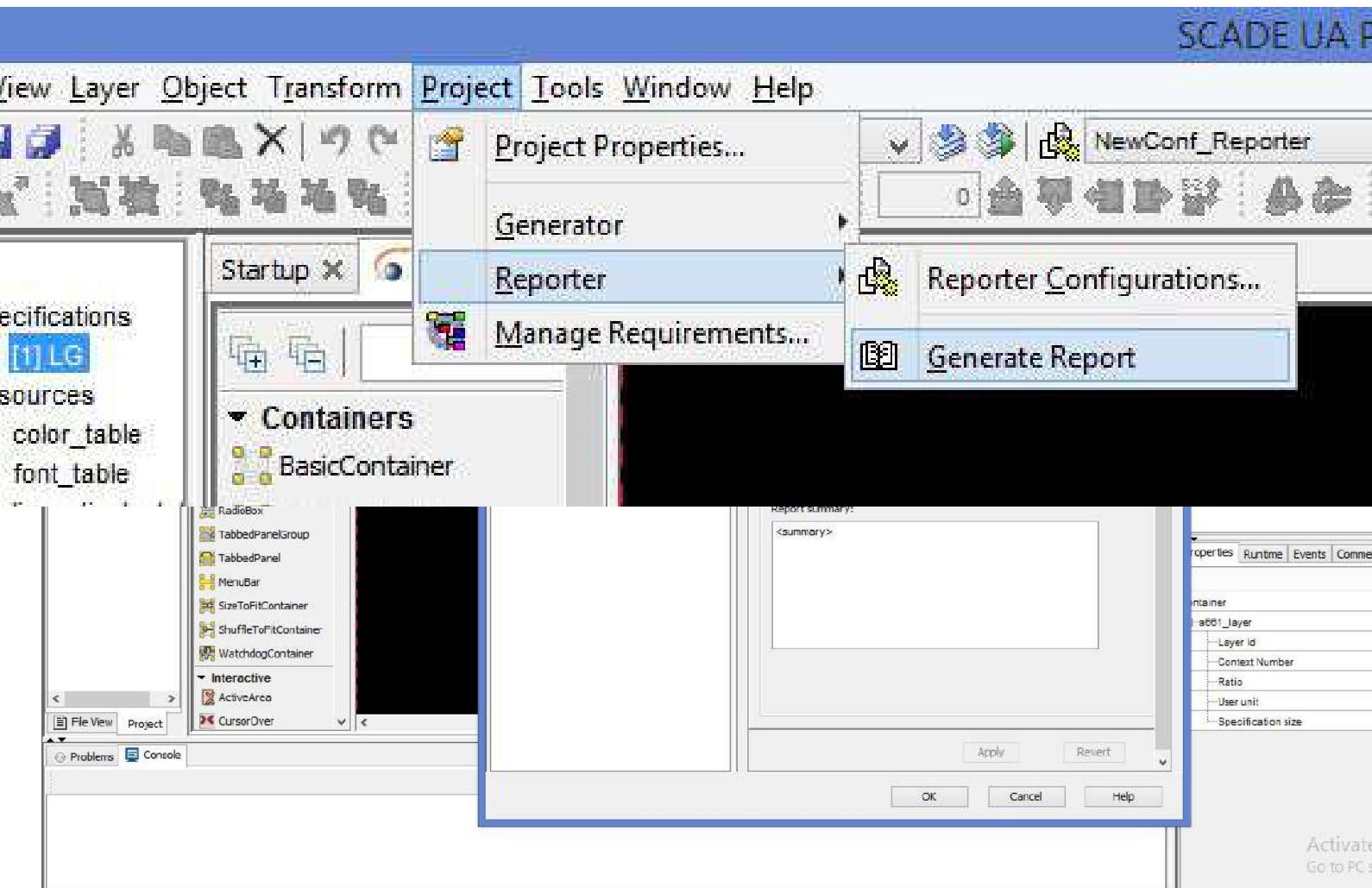
3.4 Agora, abra o *SCADE Page Creator* e abra o Definition File (DF) anteriormente elaborado, vá para *Project/Reporter/Reporter Configurations* e adicione uma nova configuração.



Preencha os campos *Project* e *Document*, conforme mostrado a seguir:



3.5 Agora, clique em *Project/Reporter/GenerateReport*



3.6 Agora, vá para a pasta do seu projeto, onde todos os documentos deverão estar, e siga o exemplo do arquivo html gerado:

'LG' SPECIFICATION

SCADE Display R17 Sun Jul 31 00:52:49 BRT 2016 | [Printable Document Version](#)

| | | | | |
|-------------------------------------|--|----------------|-----------------|------------|
| Table Of Contents | This subsection describes the information related to the pictures defined for the specification LG. Picture table: No picture defined in this specification 1.9. Symbol Table Description This subsection describes the information related to the symbol definitions defined for the specification LG. Symbol table: No symbol defined in this specification | | | |
| 1. Specification Description | <hr/> | | | |
| 1.1. General Description | | | | |
| 1.2. Specification Overview | | | | |
| 1.3. Color Table Description | | | | |
| 1.4. Line Width Table Description | | | | |
| 1.5. Line Stipple Table Description | | | | |
| 1.6. Font Table Description | | | | |
| 1.7. Layers List | | | | |
| 1.8. Picture Table Description | | | | |
| 1.9. Symbol Table Description | | | | |
| 2. Layer: a661_layer | <hr/> | | | |
| 2.1. Layer Structure | This section describes the layer 'a661_layer'. This subsection describes the structure of the layer 'a661_layer'. No hierarchy object defined. | | | |
| 2.2. Layer Content | This subsection describes the content of the layer 'a661_layer'. Object: a661_layer (type: A661Layer) <table border="1"><tr><td>Property Value</td></tr><tr><td>Visibility true</td></tr><tr><td>Layer Id 1</td></tr></table> | Property Value | Visibility true | Layer Id 1 |
| Property Value | | | | |
| Visibility true | | | | |
| Layer Id 1 | | | | |
| | Up Fail | | | |

Activate Windows
Go to PC settings to activate Windows

Mostrar todos os downloads... 

Manual SCADE 1 - E...doc Manual SCADE 1 - E...doc

Referências

Portal do Projeto SI-LANSAB. Disponível em:

<https://sites.google.com/site/projsilansab/> Acesso em 10 de julho de 2015

Portal do Projeto SI-GAC. Disponível em:

<https://sites.google.com/site/projetosigac2015/> Acesso em 10 de julho de 2015

Portal da International-pilot. Disponível em:

<http://www.international-pilot.com/first-batch-pilots-trained-embraer-legacy-500/>

Acesso em 18 de julho de 2015

MARIACHI, Samoel; JUNIOR, Carlos Lopes Nunes. WarmUp - Display ARINC 661 de um trem de pouso (LandingGear). Campo Monte Negro, 2013.

MANUAL DA EMBRAER. Treinamento SCADE ITA. 2013.

SCADE FOR DUMMIES. 2013.

Obs: As imagens dos procedimentos deste manual foram produzidas pelo próprio autor, utilizando-se da ferramenta SCADE.

<Atualizado em 04/09/2018 – Ulisses>