UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO GERADOR DE CÓDIGO ASSEMBLY PARA O MICROCONTROLADOR PIC16F84A A PARTIR DE FLUXOGRAMAS

EDUARDO SALES PINHEIRO

BLUMENAU
2004

2004/1-10

EDUARDO SALES PINHEIRO

PROTÓTIPO GERADOR DE CÓDIGO ASSEMBLY PARA O MICROCONTROLADOR PIC16F84A A PARTIR DE FLUXOGRAMAS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Miguel Alexandre Wisintainer - Orientador

BLUMENAU
2004

2004/1-10

PROTÓTIPO GERADOR DE CÓDIGO ASSEMBLY PARA O MICROCONTROLADOR PIC16F84A A PARTIR DE FLUXOGRAMAS

Por

EDUARDO SALES PINHEIRO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer – Orientador, FURB
Membro:	Prof. Antônio Carlos Tavares, FURB
Membro:	Prof. Mauro Marcelo Mattos, FURB

Blumenau, 02 de junho de 2004

Dedico este trabalho aos meus pais que me apoiaram em todos os momentos difíceis e a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste.

“Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela”.

Lina Sotis Francesco Moratti

AGRADECIMENTOS

À Deus, pelo seu imenso amor e graça.

Aos meus pais Jaime Sales Pinheiro e Salete Felsky Pinheiro que sempre estiveram presente me apoiando e me incentivando e sem os quais nada do que eu faço seria possível.

Aos meus amigos, Roque César Possamai e Luciano Sampara e a minha namorada Fabiana Russi, pelos empurrões e cobranças.

Ao meu orientador, Miguel Alexandre Wisintainer, por ter me incentivado e acreditado na conclusão deste trabalho.

RESUMO

O presente trabalho de conclusão de curso descreve a implementação de um software de suporte para simplificar a programação para microcontroladores. Apresenta-se a especificação e implementação do protótipo desenvolvido, contemplando os registradores do PIC16F84A, o tratamento de interrupções, a chamada de sub-rotinas e a interface com outros periféricos através de uma saída para escrita na serial. O funcionamento da ferramenta é caracterizado através de um estudo de caso.

Palavras chaves: Assembly; Fluxogramas; Microcontrolador PIC16F84A.

ABSTRACT

The following course´s conclusion work, describes the implements of a support software to simplify microcontrolers programming. It shows a specification and implementation of the prototype developed, contemplating the PIC16F84A's registers, the interruption's treatment, the subroutines calling and the interface with others peripherals through an output to the writting in the serial. The tool's functionalism is characterized through a case study.

Key-Words: Assembly; FlowCharts; Microcontrollers PIC16F84A.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As pessoas encontram-se rodeadas de aparelhos eletrônicos que possuem internamente um microcontrolador, e nem mesmo tem consciência disto. Videocassetes, celulares, agendas eletrônicas, vários brinquedos, alarmes de automóvel, são apenas alguns dos exemplos mais comuns (SILVA JUNIOR, 1998, p. 1).

Segundo Silva Junior (1998, p. 1), microcontrolador "é um componente que possui todos os periféricos dos microprocessadores comuns embutidos em uma só pastilha, facilitando assim o desenvolvimento de sistemas pequenos e baratos, embora complexos e sofisticados". Em resumo, um microcontrolador é um componente eletrônico utilizado para controlar um ou mais processos. E é justamente neste ponto que se insere este trabalho.

O trabalho consiste em implementar um software de suporte para simplificar a programação para microcontroladores devido ao alto grau de dificuldade encontrado durante o aprendizado da linguagem Assembly. Este trabalho complementa o trabalho desenvolvido por Fontanive (1999), que consiste num editor de fluxogramas que gera código Assembly a partir de um fluxograma. Tomando-se como base para o desenvolvimento deste protótipo, são implementados os registradores do PIC16F84A, o tratamento de interrupções, a chamada de sub-rotinas e a interface com outros periféricos através de uma saída para escrita em serial.

O protótipo de editor gráfico implementado gera a partir de um fluxograma criado pelo usuário, o código fonte Assembly em um arquivo texto para o microcontrolador PIC16F84A. A geração do código objeto para o microcontrolador é feita através do montador *MPASM* (MICROCHIP, 2003), que está integrado ao aplicativo *MPLAB* (MICROCHIP, 2004). Na realização dos testes do código gerado é utilizado o aplicativo simulador *Proteus* (ELECTRONICS, 2004).

Para implementação do protótipo foi utilizado o ambiente de programação Delphi, no qual foi instalado o componente *ExpressFlowChart* (DEVELOPER, 2004), que consiste em formas (*shapes*) e conexões no estilo de orientação a árvores e permite "modelar" o fluxo do processo do software. Este componente é responsável pela parte gráfica do protótipo (fluxograma) onde estão definidas as figuras.

Este protótipo pode ser empregado em disciplinas como Prática em Arquitetura de Computadores, Sensores e Atuadores ou Automação e Controle, com o objetivo de facilitar e agilizar o aprendizado dos acadêmicos, visto que não é necessário um estudo muito aprofundado da linguagem Assembly.

1.1 JUSTIFICATIVA DO TRABALHO

Para pessoas que estão iniciando estudos em programação de microcontroladores, a utilização de ferramentas gráficas como um editor de fluxogramas, pode tornar o estudo muito mais agradável e prático.

Pensando nestas facilidades, o trabalho mostra-se relevante no campo da ciência da computação, visto que durante seu desenvolvimento são empregadas técnicas de estruturas de dados, técnicas de programação e análise de requisitos.

Entre as contribuições que o trabalho proporciona, está o fato de que o protótipo desenvolvido pode ser empregado no meio acadêmico a fim de facilitar e agilizar o aprendizado do microcontrolador PIC16F84A através de uma ferramenta gráfica.

1.2 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo que, a partir de um fluxograma, gere código Assembly.

Os objetivos específicos do trabalho são:

a) gerar código para um conjunto de instruções em Assembly para o microcontrolador PIC16F84A;
b) permitir realizar modificações no código gerado, desvinculando este código do fluxograma;

1.3 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma:

O primeiro capítulo apresenta uma introdução sobre o assunto e objetivos do trabalho, a fim de fornecer ao leitor as informações necessárias ao entendimento do assunto abordado.

O segundo capítulo apresenta alguns conceitos e softwares que fazem parte da fundamentação teórica do trabalho proposto.

O terceiro capítulo descreve três trabalhos correlatos, sendo que o primeiro é o protótipo desenvolvido por Fontanive (1999) o qual é a base para este trabalho.

O quarto capítulo descreve a especificação e o desenvolvimento deste trabalho.

As conclusões e algumas sugestões para futuros trabalhos encontram-se no quinto capítulo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns conceitos, ferramentas e trabalhos correlatos relacionados ao projeto desenvolvido.

2.1 MICROCONTROLADORES

Segundo Souza (2002, p. 3), microcontrolador "é um pequeno componente eletrônico, dotado de uma inteligência programável, utilizado no controle de processos lógicos", onde o controle de processos é tido como o controle de periféricos, tais como: botões, *display's*, resistências e *led's*, entre outros. Já o controle de processos lógicos é tido como as operações dos sistemas que se baseiam em ações lógicas que devem ser executadas, dependendo do estado dos periféricos de entrada e/ou saída.

Os microcontroladores PIC apresentam a estrutura interna de máquina baseada na arquitetura *Harvard*, que consiste em dois barramentos internos, sendo um de dados e outro de instruções. De acordo com Souza (2002, p. 4), esse tipo de arquitetura "permite que enquanto uma instrução é executada, a outra seja "buscada" da memória, o que torna o processamento mais rápido", em contrário à grande parte dos microcontroladores tradicionais, que apresentam uma arquitetura tipo *Von-Neumam,* onde existe apenas um barramento interno para dados e instruções.

Os PIC's utilizam a tecnologia RISC que significa *Reduced Instruction Set Computer* (Computador com *Set* (jogo) de instruções reduzido). Segundo Silva Junior (1998, p. 3), esta terminologia "faz com que existam poucas instruções (mais ou menos 35, dependendo do modelo) enquanto alguns microprocessadores tradicionais chegam a ter mais de 100 instruções".

O *set* de instruções do PIC16F84A é demonstrado na Tabela 1 .

Tabela 1 – Set de instruções do microcontrolador PIC16F84A

Operações com registradores		
Instrução	Argumentos	Descrição
ANDWF	f,d	Lógica “E” entre W e f, guardando o resultado em d.
CLRF	F	Limpa f.
COMF	f,d	Calcula o complemento de f, guardando o resultado em d.
DECF	f,d	Decrementa f, guardando o resultado em d.
DECFSZ	f,d	Decrementa f, guardando o resultado em d, e pula a próxima linha se o resultado for zero.
INCF	f,d	Incrementa f, guardando o resultado em d
INCFSZ	f,d	Incrementa f, guardando o resultado em d, e pula a próxima linha se o resultado for zero.
IORWF	f,d	Lógica “OU” entre W e f, guardando o resultado em d.
MOVF	f,d	Move f para d (cópia).
MOVWF	F	Move W para f (cópia).
RLF	f,d	Rotaciona f 1 bit para esquerda.
RRF	f,d	Rotaciona f 1 bit para direita.
SUBWF	f,d	Subtrai W de f (f – W), guardando o resultado em d.
SWAPF	f,d	Executa uma inversão entre as partes alta e baixa de f, guardando o resultado em d.
XORWF	f,d	Lógico “OU exclusivo” entre W e f, guardando o resultado em d.
Operações com literais		
Instrução	Argumentos	Descrição
ADDLW	K	Soma k com W, guardando o resultado em W.
ANDLW	K	Lógica “E” entre k e W, guardando o resultado em W.
IORLW	K	Lógica “OU” entre k e W, guardando o resultado em W.
MOVLW	K	Move k para W.
SUBLW	K	Subtrai W de k (k – W), guardando o resultado em W.
XORLW	K	Lógica “OU exclusivo” entre k e W, guardando o resultado em W.
Operações com bits		
Instrução	Argumentos	Descrição
BCF	f,b	Impõe 0 (zero) ao bit do registrador f.
BSF	f,b	Impõe 1 (um) ao bit do registrador f.
BTFSC	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 0 (zero).
BTFSS	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 1 (um).
Controles		
Instrução	Argumentos	Descrição
CLRW	-	Limpa W.
NOP	-	Gasta um clico de máquina sem fazer absolutamente nada.
CALL	R	Executa a rotina R.
CLRWDT	-	Limpa o registrador WDT para não acontecer o reset.
GOTO	R	Desvia para o ponto R, mudando o PC.
RETFIE	-	Retorna de uma interrupção.
RETLW	k	Retorna de uma rotina, com k em W.
RETURN	-	Retorna de uma rotina, sem afetar W.
SLEEP	-	Coloca o PIC em modo sleep (dormindo) para economia de energia.

Fonte: Souza (2002, p. 29-30).

O PIC16F84A possui um tratador de interrupções que é controlado diretamente pelo hardware o que torna muito rápido o seu tratamento. As interrupções nos microcontroladores servem para interromper o programa a qualquer momento e está disponível em qualquer ponto do sistema. Assim sendo, quando uma interrupção ocorre, se existir um tratador de

interrupção ativo, o programa desvia para este tratador (definido pelo programador) e ao término do tratamento, o programa retorna a executar do ponto onde parou (SOUZA, 2002).

As interrupções mais comuns do PIC16F84A são:

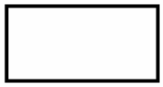
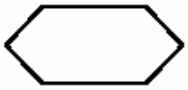
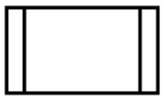
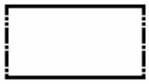
a) externa: é gerada por um sinal externo (descida de borda) ligado a uma porta específica do PIC;
b) fim de escrita na EEPROM: detecta o final de uma rotina de escrita na memória EEPROM;
c) mudança de estado: ocorre quando um sinal lógico muda de estado;
d) TMR0: ocorre sempre que um contador de tempo interno estoura.

A contagem de tempo dentro de sistemas microcontrolados é muito importante e torna possível a implementação de rotinas de pausa ao longo de um programa. Mais informações sobre os microcontroladores, o tratamento de interrupções e as rotinas de pausa podem ser vistas em Souza (2002) ou em Fontanive (1999).

2.2 FLUXOGRAMAS

Segundo Fontanive (1999, p. 5-8), fluxograma é a representação de um fluxo de dados através de uma linguagem simbólica de programação que o computador interpreta. Ao se deparar com um problema, o programador trata de resolvê-lo, subdividindo-o em duas fases inter-relacionadas. Em primeiro lugar, faz o diagrama de blocos (fluxograma) logo após ter estudado o problema. Em segundo lugar, codifica-o, isto é, transforma o diagrama de blocos em uma linguagem que o computador compreenda.

De acordo com Cares (2002, p. 15-16), o fluxograma foi um dos primeiros e mais utilizados métodos para diagramação, tendo sido aceito pela maioria dos programadores e analistas de sistemas antes do surgimento dos métodos de solução estruturada. É uma ferramenta gráfica, que permite representar qualquer tipo de solução de problema (simples ou complexos) e é composto por um conjunto de símbolos pré-definidos representados pela Figura 1.

	**Terminal**	Indica início e fim do fluxo do programa.
	**Seta de fluxo de dados**	Indica o sentido do fluxo de dados. Serve para conectar os blocos existentes
	**Entrada de dados via teclado**	Indica que serão recebidas informações através do teclado.
	**Processamento**	Indica a realização de cálculos, atribuições ou qualquer manipulação de dados.
	**Exibir no display (tela)**	Indica que as informações serão exibidas através do display.
	**Desvio Condicional**	Indica tomada de decisão. Divide o fluxo do programa em dois caminhos.
	**Conector**	Serve para conectar os caminhos que foram desviados.
	**Repetição com variável de controle**	Usado especificamente para o comando de repetição: PARA<>DE<>ATE<>PASSO<>
	**Subrotina**	Usado para indicar a chamada a um subprograma (procedimento ou função).
	**Declaração**	Usado para a declaração das variáveis, tipos e constantes.

Fonte: Cares (2002, p. 16)

Figura 1 – Representação dos objetos de um fluxograma

Na implementação do protótipo faz-se uso de uma notação de fluxograma um pouco diferente da notação tradicional, isto porque o componente utilizado na implementação possui algumas limitações quanto à forma dos objetos que são criados. Este componente possui apenas objetos com o formato de retângulos, triângulos, elipses e hexágonos.

Por exemplo, para representar um conector no fluxograma faz-se uso da figura que representa um triângulo e o bloco de entrada de dados é dado por uma figura em forma de retângulo.

Portanto, tomando como base estas definições, o protótipo aqui implementado parte do princípio de que o trabalho do programador será menos tedioso quando o mesmo passar a utilizar técnicas visuais (através de fluxogramas) para desenvolver seu software.

2.3 GRAFOS

Um grafo consiste num conjunto de **nós** (ou vértices) e num conjunto de **arcos** (ou arestas) onde, cada arco num grafo é especificado por um par de nós e é representado por setas (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

Os grafos possuem diversos métodos e algoritmos de percurso (busca). A seguir serão vistos dois métodos (busca em profundidade e largura), sendo que o método de busca em profundidade foi utilizado para percorrer o fluxograma gerado pelo protótipo.

O método de busca em profundidade, também chamado de "*depth-first*", consiste em explorar preferencialmente os "descendentes" de um nó que tenha sido avaliado em prejuízo dos "vizinhos" ou "irmãos" deste nó. O algoritmo de busca em profundidade pode ser visto no Quadro 1.

```
CRIE DUAS PILHAS ABERTOS E FECHADOS
INICIALIZE A PILHA ABERTOS = [NÓ INÍCIO]
ENQUANTO ABERTOS NÃO ESTIVER VAZIA FAÇA
   REMOVA O NÓ DO TOPO DE ABERTOS E CHAME-O DE X
   SE X É CONCLUSIVO TERMINE COM SUCESSO
   SENÃO BUSQUE TODOS OS FILHOS DE X
      DISPENSE OS FILHOS DE X QUE JÁ ESTÃO EM ABERTOS OU FECHADOS
      COLOQUE SOBRE A PILHA ABERTOS OS FILHOS REMANESCENTES DE X (NA ORDEM EM
      QUE FORAM BUSCADOS??)
   COLOQUE X SOBRE A PILHA FECHADOS
FIM ENQUANTO
```

Quadro 1 – Algoritmo de busca em profundidade

O método de busca em largura, em contraste com o método da busca em profundidade, consiste em explorar os nós nível-a-nível. Neste caso são priorizados os "vizinhos" ou "irmãos" de um nó que tenha sido avaliado em prejuízo dos "descendentes". O algoritmo de busca em largura pode ser visto no Quadro 2.

```
CRIE DUAS FILAS ABERTOS E FECHADOS
INICIALIZE A FILA ABERTOS = [NÓ INÍCIO]
ENQUANTO ABERTOS NÃO ESTIVER VAZIA FAÇA
   REMOVA UM NÓ DA FILA ABERTOS E CHAME-O DE X
   SE X É CONCLUSIVO TERMINE COM SUCESSO
   SENÃO BUSQUE TODOS OS FILHOS DE X
      DISPENSE OS FILHOS DE X QUE JÁ ESTÃO EM ABERTOS OU FECHADOS
      COLOQUE NA FILA ABERTOS OS FILHOS REMANESCENTES DE X (NA ORDEM EM QUE
      FORAM  BUSCADOS??)
   COLOQUE X NA FILA FECHADOS
FIM ENQUANTO
```

Quadro 2 – Algoritmo de busca em largura

Nos grafos o termo **caminho** significa uma seqüência de um ou mais arcos em que o segundo nó de cada arco coincide com o primeiro do seguinte, permitindo atingir um nó B a partir de um nó A.

Os grafos ainda podem ser considerados conexos se todo nó no grafo for atingível a partir de qualquer outro nó ou biconexos se não possuir pontos de articulação.

Em analogia aos conceitos de grafos, o fluxograma desenvolvido a partir do protótipo é um grafo pois faz uso diversos caminhos e todos os nós (blocos) possuem pelo menos uma conexão e isto o torna em grafo conexo.

Mais informações sobre grafos podem ser obtidas em (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

2.4 EXPRESSFLOWCHART SUITE

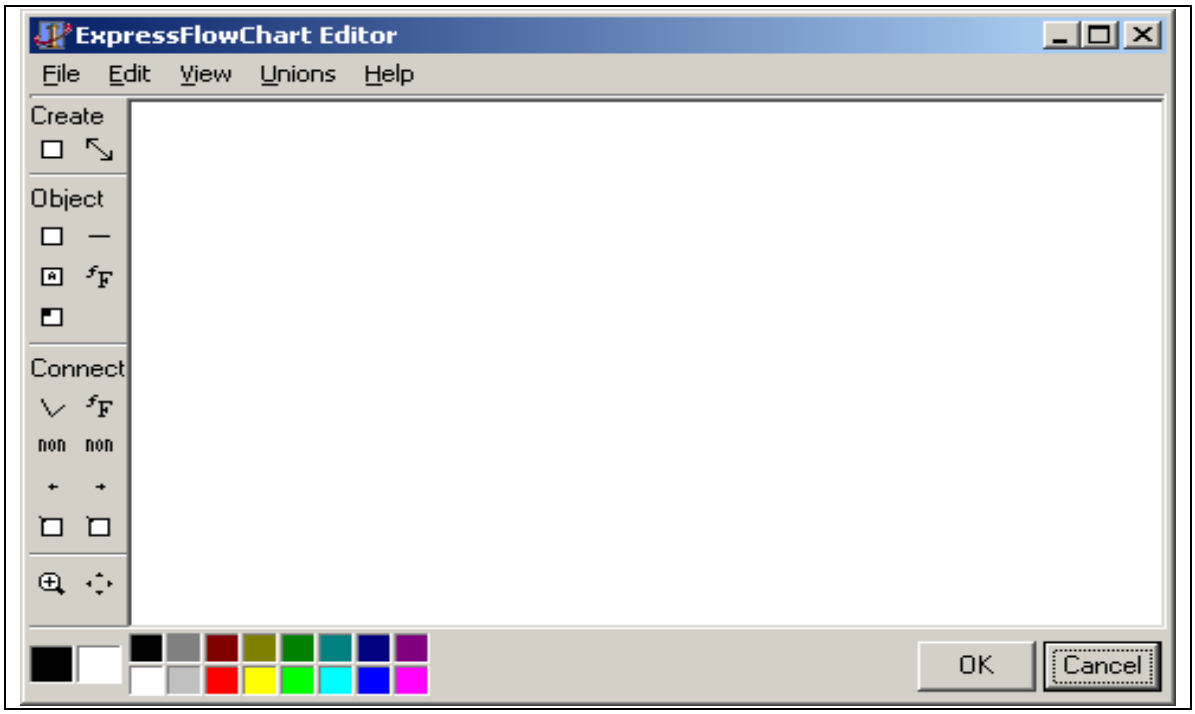
Segundo Developer (2004), o *ExpressFlowChart Suite* é uma ferramenta que permite a fácil criação de aplicações que contenham quadros, esquemas, hierarquias, gráficos e foi utilizada na construção do protótipo.

No ambiente de desenvolvimento desta ferramenta existe um controle de fluxograma onde são criados os objetos e as conexões. Um bloco dentro deste controle é chamado de

objeto. Os objetos podem ser de formas, cores e tamanhos diferentes, além de possuir ou não um texto ou uma figura associada.

Os objetos do fluxograma são ligados através de conexões, onde cada conexão pode ligar apenas dois objetos e pode ser desenhada de maneiras diferentes. Por exemplo, uma conexão pode ser representada por uma reta ou uma curva entre dois objetos.

O ambiente do *ExpressFlowChart* pode ser visto na Figura 2.



Fonte: Developer (2004).

Figura 2 – Ambiente do *ExpressFlowChart Suite*

2.4.1 Propriedades e métodos mais utilizados

Na Tabela 2 são relatados alguns dos eventos, métodos e propriedades do *ExpressFlowChart* mais utilizados e mais importantes para a implementação do protótipo. Alguns exemplos da utilização destes métodos e propriedades podem ser vistos capítulo 4.

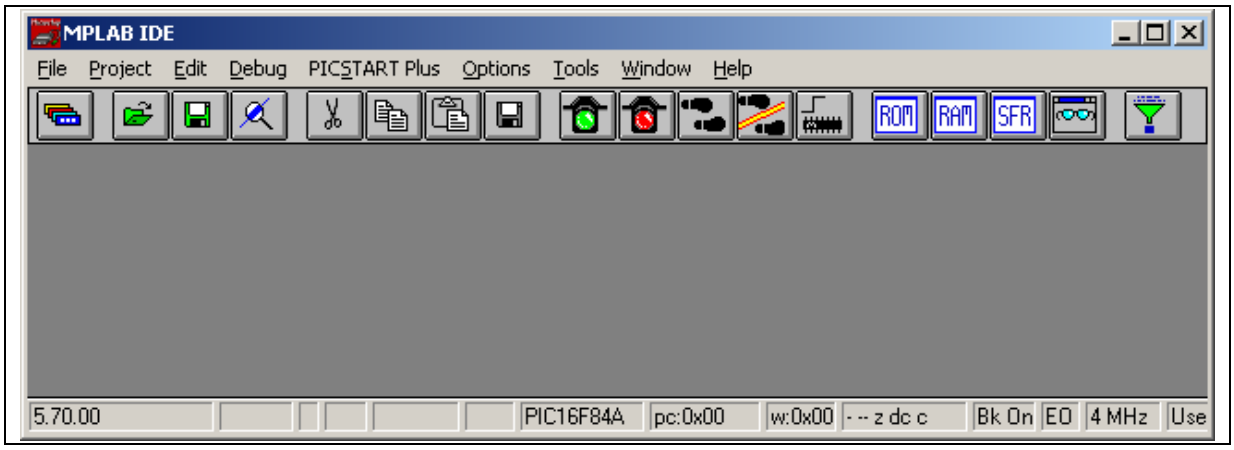
Tabela 2 – Propriedades e métodos do ExpressFlowChart Suite

TdxCustomFlowChart	
Propriedade/Método/Evento	**Descrição**
Propriedade Connections	Representa uma lista de todas as conexões de um fluxograma.
Propriedade LeftEdge	Determina a posição à esquerda de um objeto nas coordenadas do controle do fluxograma.
Propriedade ObjectCount	Retorna o número total de objetos dentro de um fluxograma.
Propriedade Objects	Representa uma lista de todos os objetos de um fluxograma.
Propriedade SelectedConnection	Retorna à conexão selecionada.
Propriedade SelectedConnectionCount	Retorna à quantidade de conexões selecionadas.
Propriedade SelectedConnections	Representa uma lista de todas as conexões selecionadas em um fluxograma.
Propriedade SelectedObject	Retorna o objeto selecionado.
Propriedade SelectedObjectCount	Retorna à quantidade de objetos selecionados.
Propriedade SelectedObjects	Representa uma lista de todos os objetos selecionados em um fluxograma.
Propriedade LeftEdge	Determina à posição superior de um objeto nas coordenadas do controle do fluxograma.
Método CreateConnection	Cria uma nova conexão.
Método CreateObject	Cria um novo objeto.
Método DeleteConnection	Exclui uma conexão específica.
Método DeleteObject	Exclui um objeto específico.
Método LoadFromFile	Recupera objetos do fluxograma de um arquivo especificado por FileName.
Método SaveToFile	Salva o fluxograma em um arquivo especificado por FileName.
Evento OnChange	Ocorre ao mudar um item dentro de um fluxograma.
Evento OnDeletion	Ocorre quando um item estiver a ponto de ser excluído.
Evento OnSelection	Ocorre quando um item estiver a ponto de ser selecionado.
TdxFcConnection	
Propriedade/Método	**Descrição**
Propriedade ObjectSource	Especifica um objeto de origem para uma conexão.
Propriedade PenStyle	Especifica o estilo da linha para uma conexão.
Método Create	Cria uma instância do tipo TdxFcConnection.
Método Destroy	Destrói uma instância do tipo TdxFcConnection e libera a memória alocada.
TdxFcItens	
Propriedade/Método	**Descrição**
Propriedade Font	Determina características da fonte do texto de um objeto.
Propriedade Text	Representa uma string de texto associada a um objeto.
Método Create	Cria uma instância do tipo TdxFcItem.
Método Destroy	Destrói uma instância do tipo TdxFcItem e libera toda a memória alocada.
TdxFcObjects	
Propriedade/Método	**Descrição**
Propriedade CustomData	Especifica um nome a um objeto.
Propriedade Height	Determina a altura de um objeto.
Propriedade HorzImagePos	Determina a posição horizontal de um objeto.
Propriedade Left	Determina a coordenada a esquerda de um objeto do fluxograma.
Propriedade ObjectCount	Determina o número de objetos filhos para um dado objeto.
Propriedade Objects	Representa uma lista de objetos ligados ao objeto principal.
Método Create	Cria uma instância do tipo TdxFcObject.
Método Destroy	Destrói uma instância do tipo TdxFcObject e libera toda a memória alocada.

Fonte: adaptado de Developer (2004).

2.5 MPLAB

O *MPLAB* é um ambiente de desenvolvimento para programas PICs que roda na plataforma Windows. Segundo Souza (2002, p. 31) o *MPLAB* "junta num mesmo ambiente, o gerenciamento de projetos, a compilação, a simulação, a emulação e a gravação" do PIC. A imagem deste ambiente pode ser vista na Figura 3.



Fonte: Microchip (2004)

Figura 3 – Ambiente do *MPLAB*

Para gerar o código objeto através do *MPLAB* a partir de um código fonte gerado pelo protótipo, é necessário que se crie inicialmente um **projeto** que é composto por um arquivo que guarda as informações necessárias ao sistema em desenvolvimento. Após criar um projeto, deve-se associar a ele um arquivo com extensão "asm" gerado pelo protótipo. Depois, deve-se montar este arquivo associado anteriormente, gerando assim um arquivo hexadecimal contendo o código objeto do programa. Mais informações sobre o *MPLAB* podem ser vistas em Microchip (2004).

Tal ferramenta é utilizada neste trabalho para gerar o código objeto a partir do arquivo gerado pelo protótipo que contém o código do programa em Assembly. Com o código objeto gerado é possível realizar os testes de validação do protótipo.

2.6 PROTEUS

O *Proteus* é um aplicativo simulador interativo. Com ele é possível criar e simular o funcionamento de circuitos eletrônicos sem a necessidade de se criar um circuito físico. Com isto, o tempo de construção e adaptação se torna muito menor (ELECTRONICS, 2004).

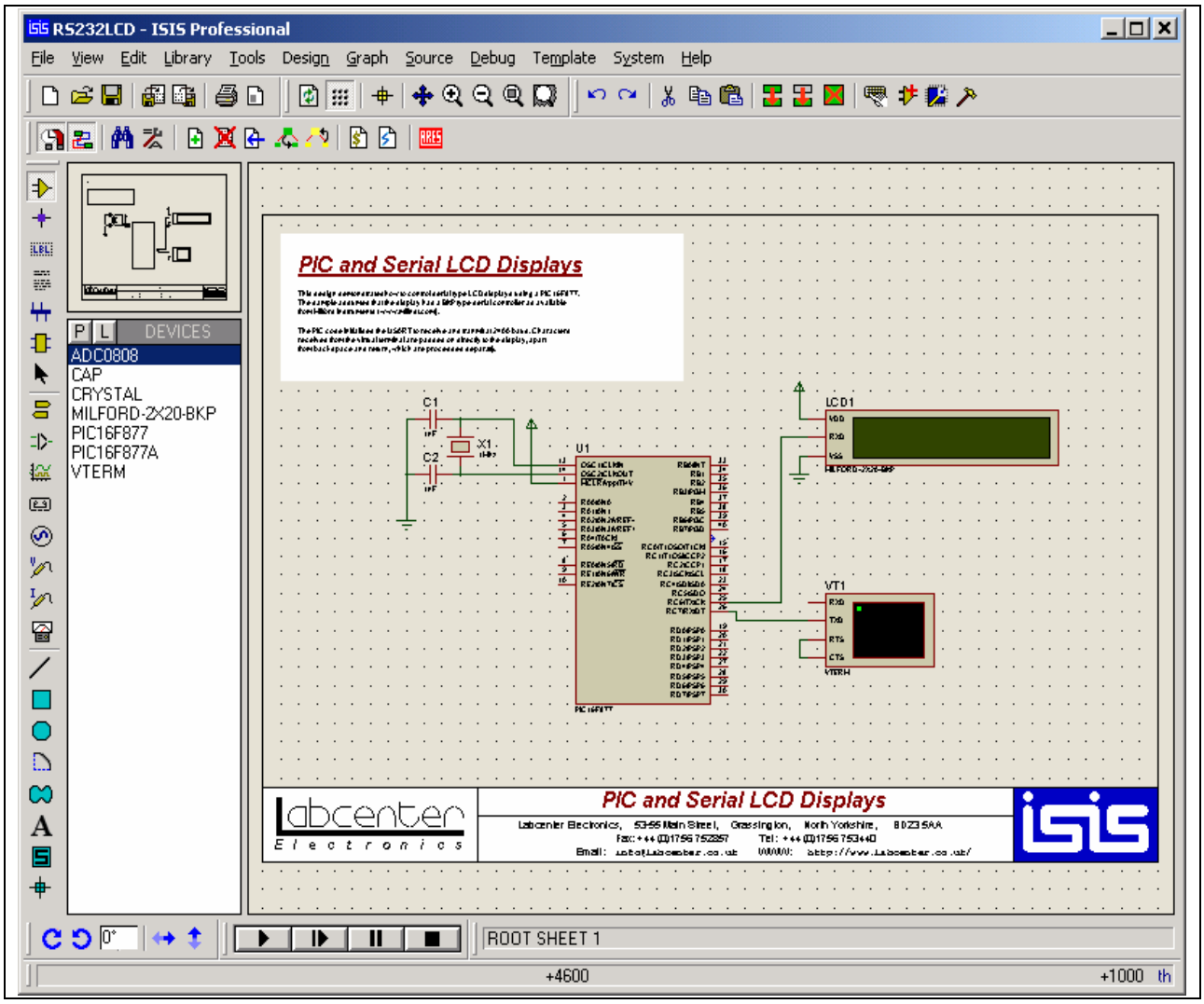
Segundo Electronics (2004), através do *Proteus* é possível criar um circuito completo para um sistema baseado em microcontrolador e testá-lo interativamente, tudo dentro de um mesmo software. Para construção dos circuitos, o *Proteus* disponibiliza uma ampla variedade de componentes eletrônicos como os microcontroladores, os *led's*, as portas lógicas, etc.

Entre as características do *Proteus* pode-se citar:

a) capacidade de importar figuras e outros componentes;
b) capacidade de simular várias famílias de microcontroladores;
c) capacidade de selecionar objetos e atribuir suas propriedades;
a) fornecimento de suporte a circuitos secundários, que podem ser ligados ao circuito principal;
b) fornecimento de suporte total a barramentos incluindo pinos, terminais, portas e fios;
c) incorporação de suporte à gerência de grandes projetos, controlando milhares de componentes;
d) numeração automática dos componentes;
e) possibilidade de personalizar cores e fontes dos objetos;
f) trabalha com instrumentação, osciloscópios, etc.

Portanto, com base em algumas características e funcionalidades deste ambiente, em especial a possibilidade de se criar circuitos utilizando diversas famílias de microcontroladores, este ambiente demonstra capacidade para simular o software (código objeto) gerado a partir do protótipo desenvolvido. O circuito criado para simular o programa gerado pelo protótipo pode ser visto no estudo de caso desenvolvido no capítulo 4.

Na Figura 4 pode ser visto o ambiente de desenvolvimento e simulação do aplicativo *Proteus*.



Fonte: Electronics (2004)

Figura 4 – Ambiente de simulação do *Proteus*

2.7 EDITOR DE FLUXOGRAMAS PARA GERAÇÃO DE CÓDIGO ASSEMBLY

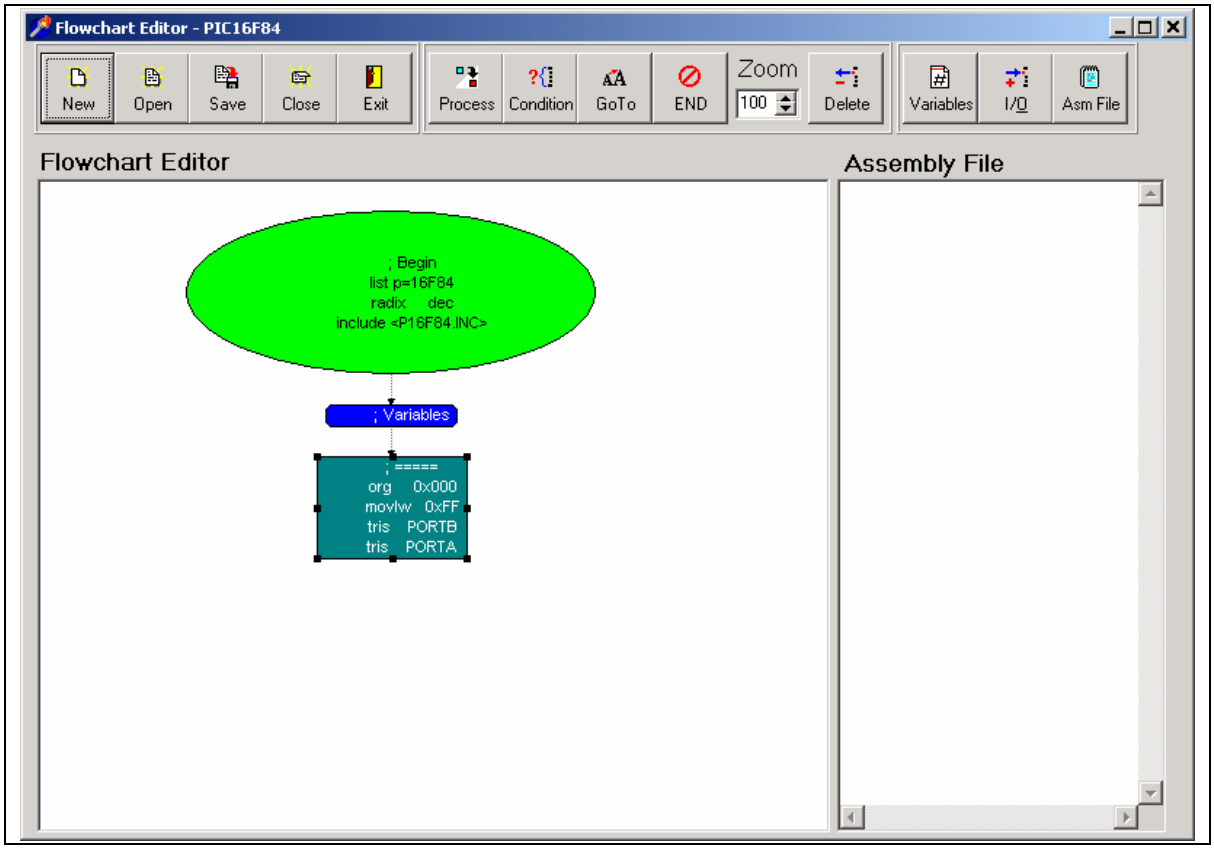
O trabalho correlato aqui descrito refere-se a um editor de fluxogramas desenvolvido por Fontanive (1999), e tem o mesmo objetivo do protótipo aqui desenvolvido. O software em questão gera o código Assembly a partir de um fluxograma, porém este software possui algumas restrições e limitações que influenciam muito no desenvolvimento de um bom programa para um microcontrolador.

Algumas limitações e falhas detectadas neste software são:

a) não possui a instrução de **interrupção** que é essencial para o desenvolvimento de uma boa aplicação;
b) ocorrem algumas violações de acesso que inviabilizam o uso contínuo do protótipo;
c) não permite que o usuário escolha o local onde será salvo o fluxograma e o código Assembly;
d) não possui a instrução de **pausa** que é essencial para o controle de algumas ações.

Neste editor, o usuário cria o fluxograma através das opções disponíveis na barra de botões que está localizada na parte superior da tela. Ao término da edição do fluxograma o usuário deve pressionar no botão "Asm File" para compilar o fluxograma e gerar o código Assembly que é apresentado no *memo* "Assembly File".

A tela principal do protótipo pode ser vista na Figura 5.



Fonte: Fontanive (1999)

Figura 5 – Tela principal do protótipo desenvolvido por Fontanive (1999)

2.8 CHBASIC

Segundo Ezimerchant (2003), o *CH Basic - Flash Edition* é uma ferramenta que simplifica o desenvolvimento e escrita de programas para os microcontroladores PIC da família 16Fxxx. Este programa contém um conjunto básico de instruções (em Basic) e possui componentes visuais em forma de caixa de diálogo que auxiliam e contribuem para reduzir o tempo de desenvolvimento de um software para o PIC.

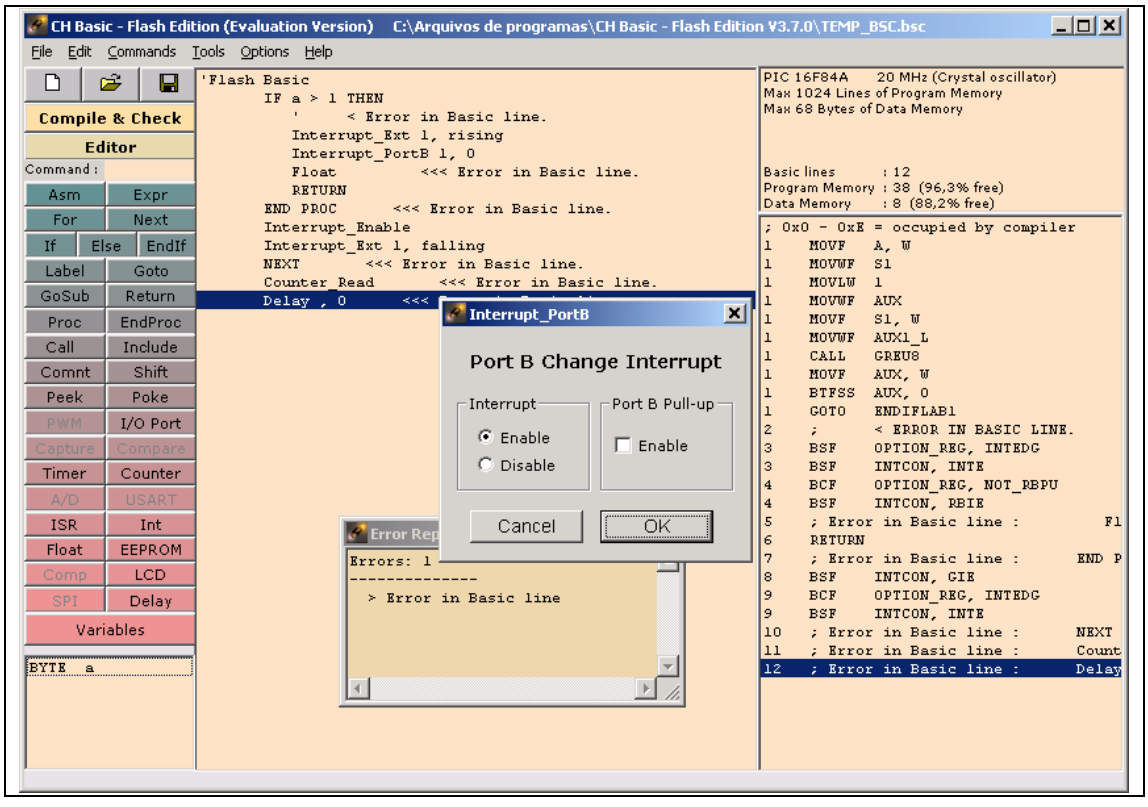
Usando o *CH Basic* como ferramenta para escrever programas para microcontroladores, tem-se alguns benefícios como:

a) grande redução do tempo que leva para escrever programas para microcontroladores PIC;

b) não é necessário que o programador aprenda a linguagem Assembly para desenvolver seus programas;

c) possui comandos simples e intuitivos.

A tela principal do *CH Basic* pode ser vista na Figura 6.



Fonte: Ezimerchant (2003)

Figura 6 – Tela principal do *CH Basic*

Este ambiente foi utilizado para verificar as estruturas dos comandos gerados na linguagem Assembly e que foram implementadas no protótipo aqui desenvolvido.

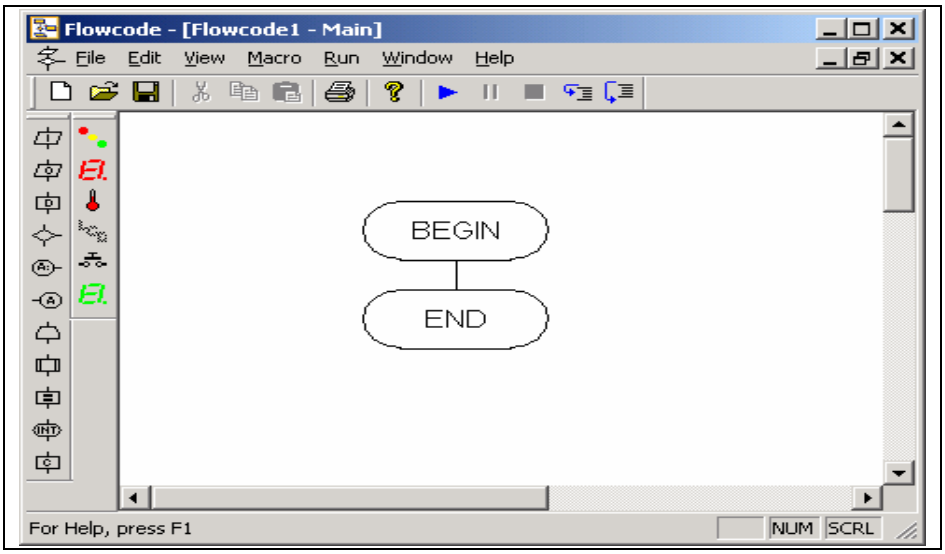
2.9 FLOWCODE

Segundo Multimídia (2004), *Flowcode* é um software que permite criar programas complexos para microcontroladores PIC. Com este software pode-se criar fluxogramas e simular a sua execução passo a passo. Além desta funcionalidade, é possível gerar código objeto para PIC e/ou gerar código Assembly em arquivo texto, tudo isto a partir do fluxograma desenvolvido pelo usuário.

Quando o usuário cria um novo fluxograma, o sistema gera automaticamente um nó inicial e um nó final. A partir deste ponto, o usuário deve arrastar os componentes que estão localizados ao lado esquerdo da tela e inserir no ponto desejado do fluxograma.

Para alguns blocos é necessário editar o código Assembly que está nas suas propriedades. Para isto, o usuário deve dar um duplo clique com o mouse sobre o objeto desejado e em seguida deve alterar o código que já está inserido. Seguindo este procedimento é possível criar o fluxograma desejado. Após criar o fluxograma, o usuário pode executá-lo passo a passo.

A tela principal do aplicativo *FlowCode* pode ser vista na Figura 7.



Fonte: Multimidia (2004)

Figura 7 – Tela principal do *FlowCode*

3 DESENVOLVIMENTO DO TRABALHO

O presente capítulo descreve a especificação, a implementação e os testes do protótipo que gera código Assembly a partir de fluxogramas. Na primeira seção estão descritos os requisitos do protótipo. A segunda seção mostra a especificação do protótipo através de casos de uso e diagramas de atividades. A terceira seção descreve a implementação, as técnicas, as ferramentas utilizadas e a operacionalidade da implementação. Na quarta e última seção estão descritos os resultados obtidos e as discussões referentes ao trabalho correlato desenvolvido por Fontanive (1999) descrito na fundamentação teórica.

3.1 REQUISITOS PRINCIPAIS

Nesta seção estão descritos os requisitos principais do protótipo. Estes requisitos são parte essencial para que seja alcançado o objetivo esperado.

Para isto, o protótipo deve ser capaz de:

a) criar dois arquivos de saída, um contendo o fluxograma e outro contendo o código gerado (Requisito Funcional - RF);
b) disponibilizar recursos para trabalhar com registradores, variáveis e literais (RF);
c) gerar código Assembly a partir do fluxograma, respeitando as ligações definidas pelo usuário (RF);
d) permitir realizar modificações no código gerado, desvinculando este código do fluxograma (RF);
e) possuir uma interface simples e de fácil uso, seguindo a especificação das janelas e menus do Windows (Requisito Não Funcional - RNF);
f) verificar ligações entre os componentes gráficos do fluxograma durante sua construção (RF).

3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do protótipo gerador do código Assembly a partir de fluxogramas, sendo apresentados os diagramas da linguagem de modelação visual UML (*Unified Modeling Language*) utilizados para especificar a estrutura do protótipo através do uso da ferramenta *SmartDraw* (HEMERA, 2004). O primeiro diagrama utilizado

na especificação é o de caso de uso. Na seqüência, o protótipo é representado na forma de diagramas de atividades.

3.2.1 Diagramas de casos de uso

O diagrama de casos de uso tem como propósito principal descrever de forma conceitual a estrutura do protótipo (FURLAN, 1998). A Figura 8 mostra o diagrama de caso de uso do protótipo, visto do ponto de vista do usuário que utiliza o sistema.

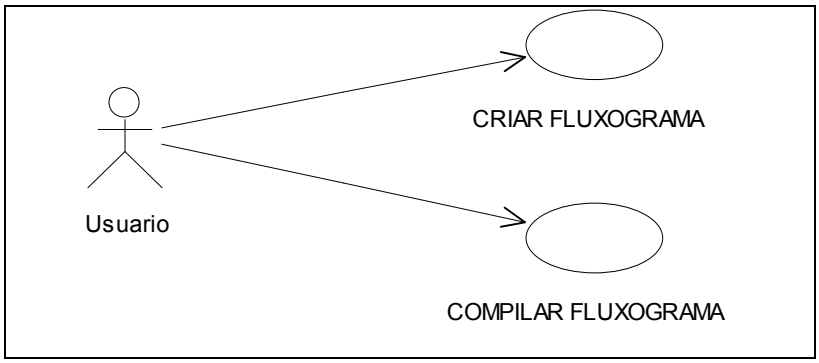


Figura 8 – Apresentação do Diagrama de Casos de Uso

O Quadro 3 descreve cada um dos casos de uso do protótipo, indicando o nome do caso de uso, o respectivo ator e uma descrição resumida de cada um deles.

CRIAR FLUXOGRAMA

Sumário: O usuário usa a ferramenta (protótipo) para resolver um problema de estudo de caso retornando ao fluxograma ou pode criar um novo.

Ator Principal: Usuário

Precondições: O fluxograma terá que obedecer às restrições definidas de ligações entre os objetos.

Fluxo Principal:

a) O usuário abre ou cria um fluxograma.
b) O fluxograma editado pode ser salvo ou alterado.

COMPILAR FLUXOGRAMA

Sumário: O usuário inicia com o processo de compilar o fluxograma.

Ator Principal: Usuário

Precondições: Deve estar com o fluxograma gerado completamente sem sua estrutura corrompida.

Fluxo Principal:

a) O usuário clica no botão compilar.
b) O usuário deve definir uma porta do microcontrolador que será usada pelo objeto que indica uma serial quando solicitado pelo software.

Quadro 3 - Descrição dos casos de uso

3.2.2 Diagramas de atividades

Nesta seção estão demonstrados os diagramas de atividades das principais rotinas do protótipo. O primeiro diagrama representado pela Figura 9 mostra a rotina para se incluir um bloco de pausa no fluxograma.

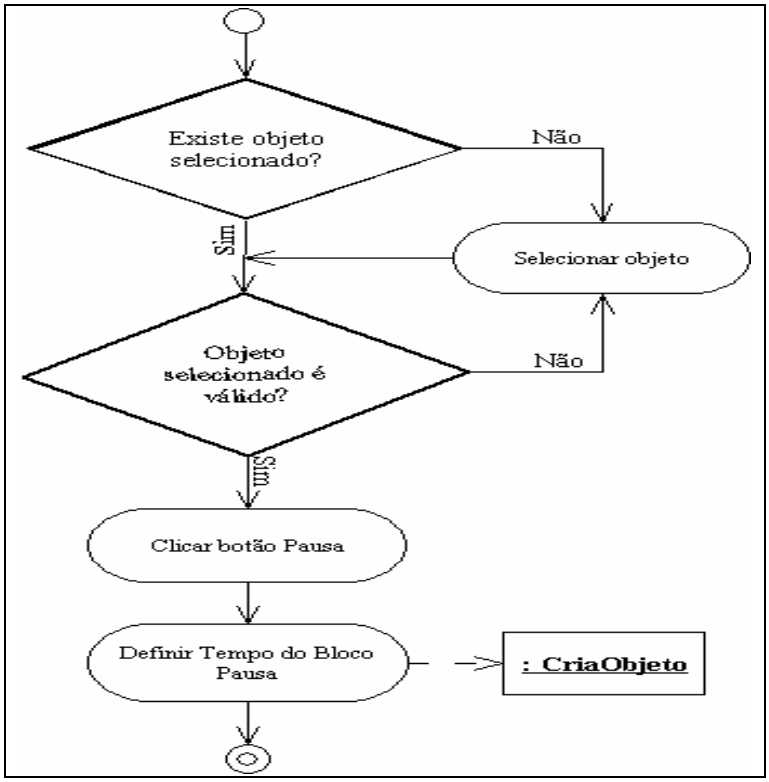


Figura 9 – Diagrama de atividades – Rotina de Pausa

A rotina de condição pode ser vista no diagrama representado na Figura 10.

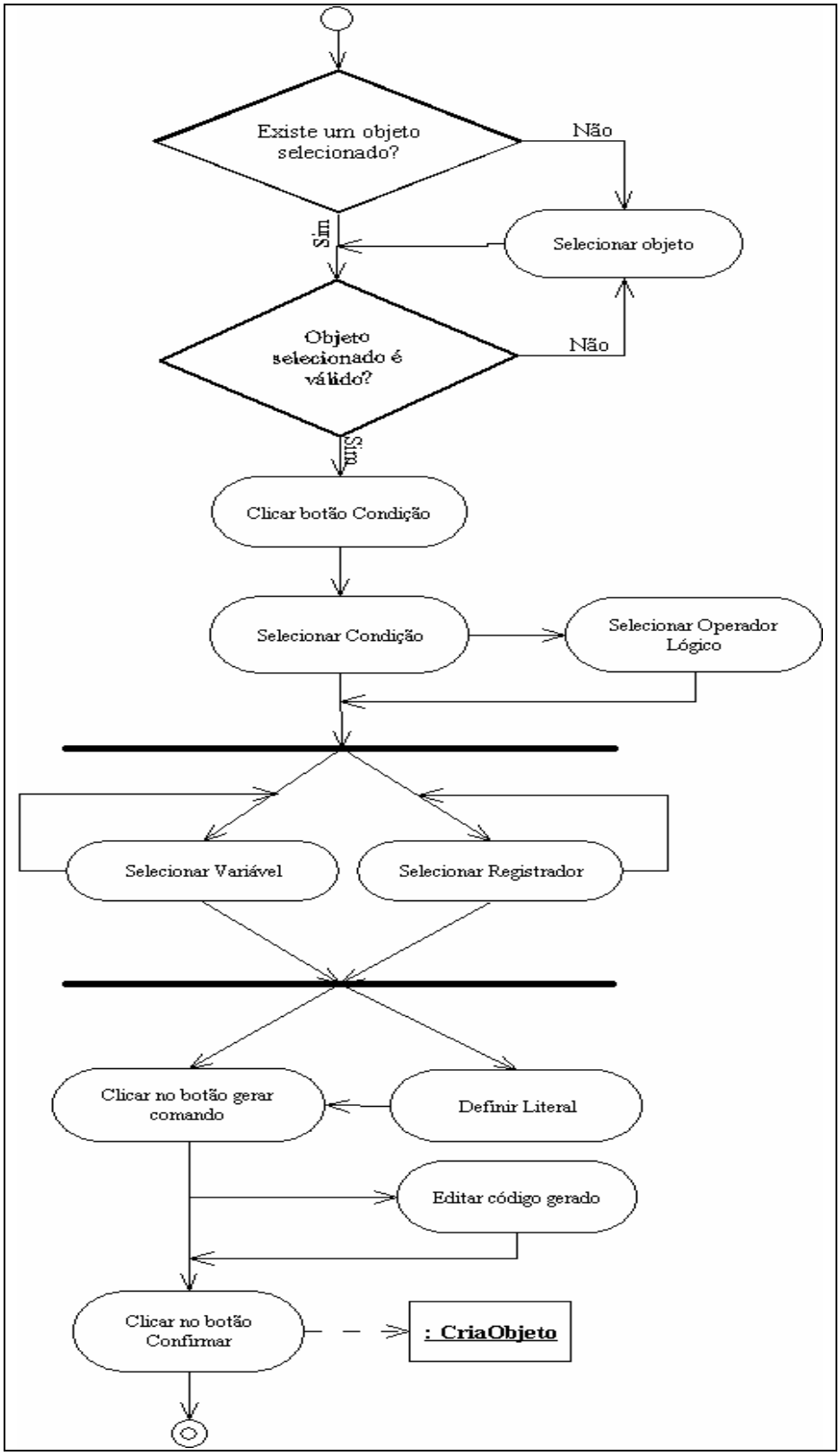


Figura 10 - Diagrama de atividades – Rotina de Condição

A rotina de processos pode ser vista no diagrama representado na Figura 11.

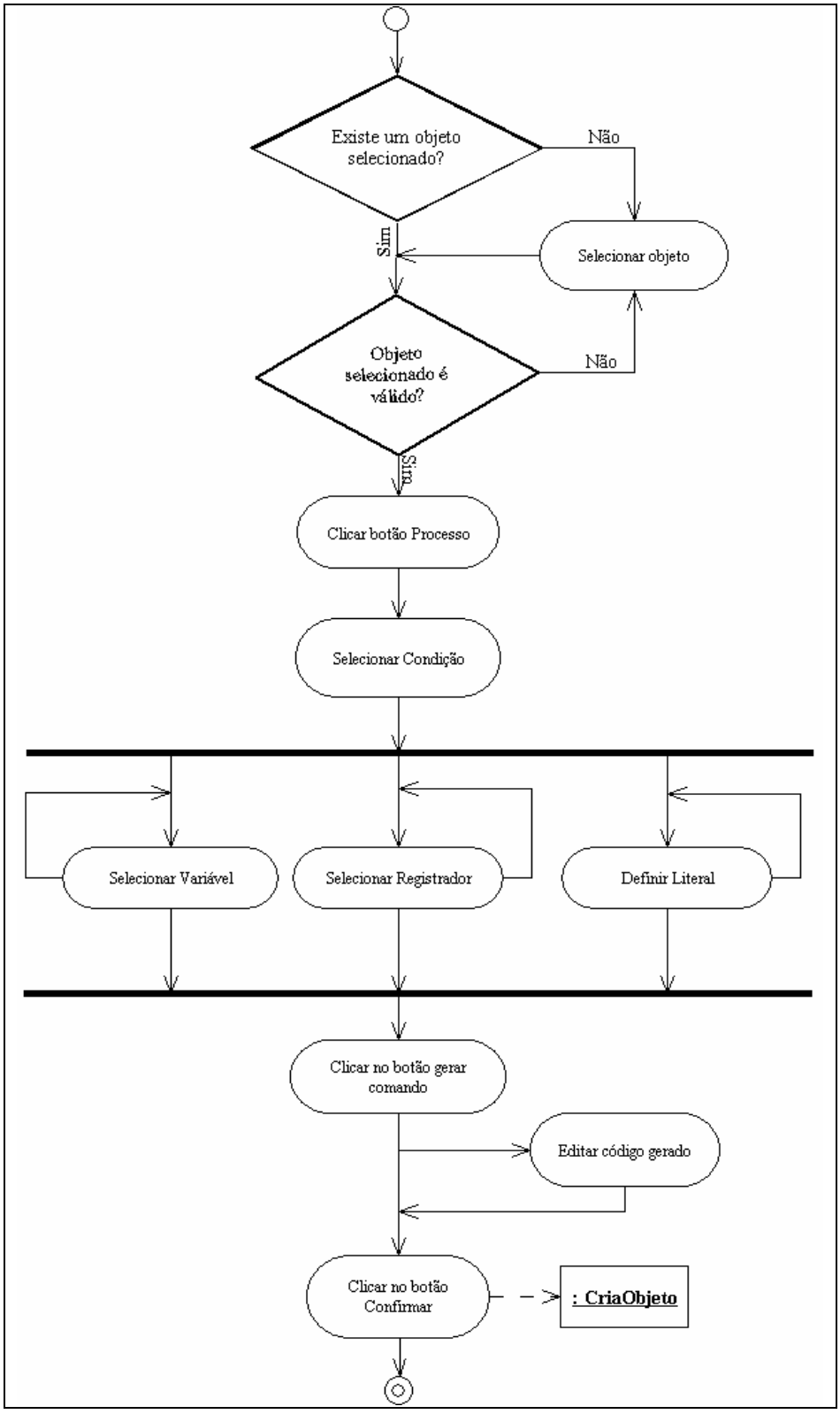


Figura 11 - Diagrama de atividades – Rotina de Processo

A rotina de *goto* pode ser vista no diagrama representado na Figura 12.

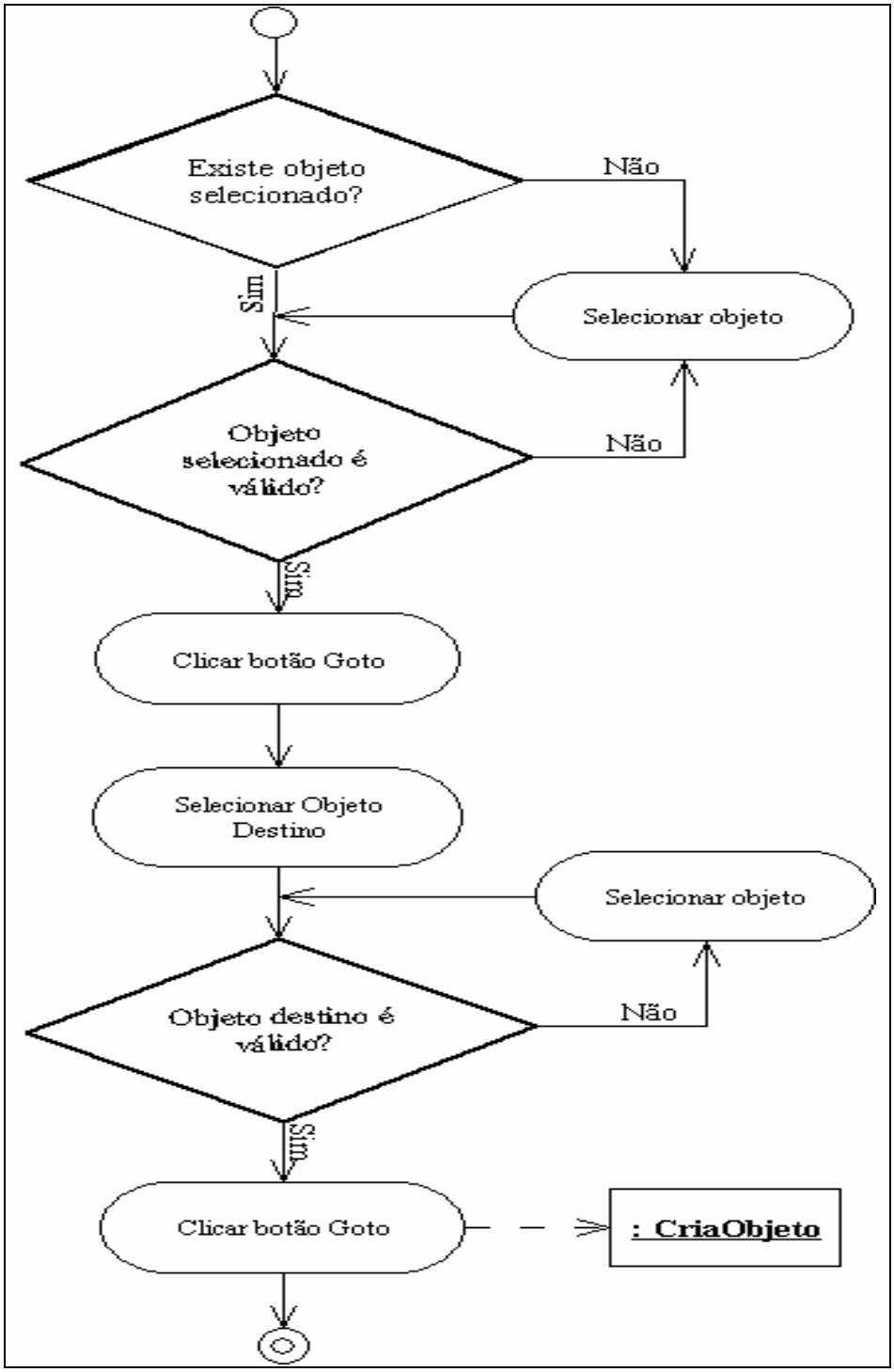


Figura 12 - Diagrama de atividades – Rotina de Goto

A rotina de serial pode ser vista no diagrama representado na Figura 13.

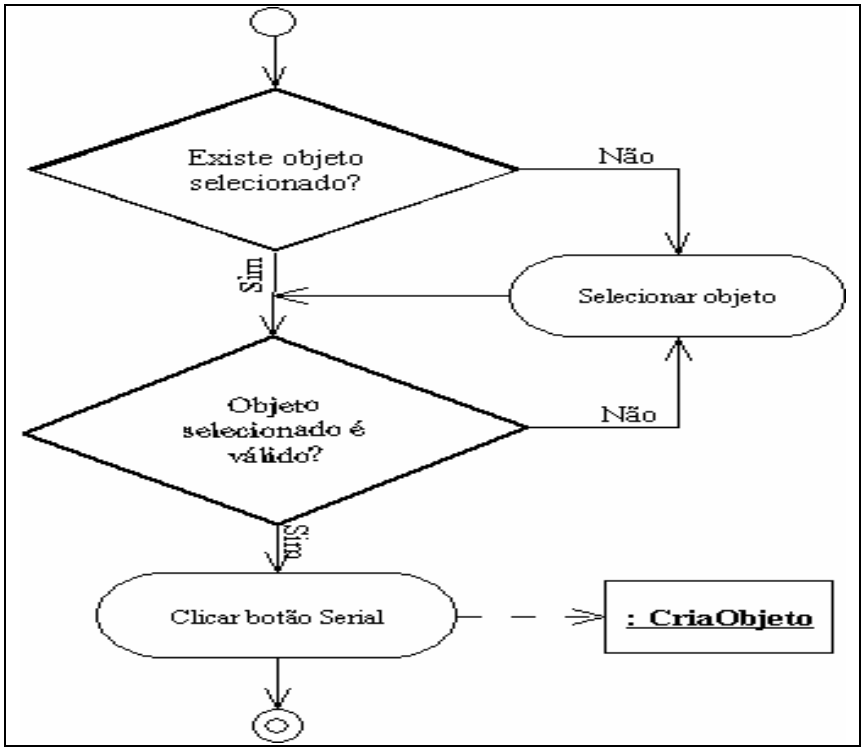


Figura 13 - Diagrama de atividades – Rotina de Serial

A rotina de fim pode ser vista no diagrama representado na Figura 14.

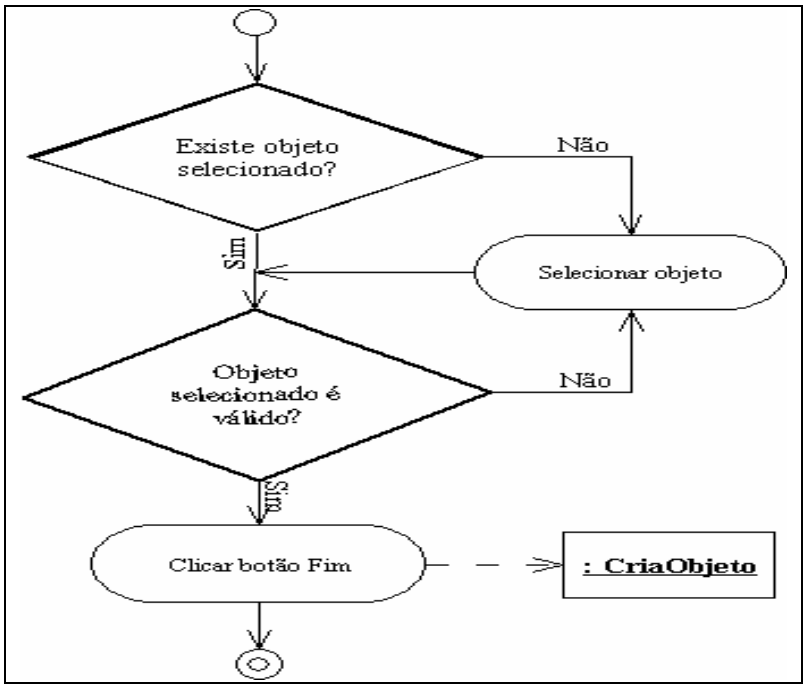


Figura 14 - Diagrama de atividades – Rotina de Fim

3.3 IMPLEMENTAÇÃO

Nesta seção mostra-se detalhada a implementação do protótipo, onde são descritas as técnicas e ferramentas utilizadas e a operacionalidade da implementação através de um estudo de caso. Para finalizar, são comentados os resultados obtidos a partir deste estudo de caso.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

No desenvolvimento da aplicação é utilizado o ambiente de desenvolvimento Delphi 5. Neste ambiente instala-se o componente (adquirido) *ExpressFlowChart* (DEVELOPER, 2004) que é responsável por criar todos os objetos dos fluxogramas. Um exemplo do código que cria um objeto no fluxograma pode ser visto no Quadro 4.

No Quadro 4 é possível identificar algumas propriedades e métodos do componente *ExpressFlowChart* (DEVELOPER, 2004), que foi estudado no capítulo 2 e utilizados na contrução dos fluxogramas. Alguns exemplos são: *"CreateConnection", "CreateObject", "ObjectCount"* e *"SelectedObject"*.

Para especificação do protótipo utiliza-se a ferramenta *SmartDraw* (EMERA, 2004), onde são construídos os diagramas de casos de uso e os diagramas de atividades.

Para montagem do código Assembly (compilar o fluxograma) é utilizado um algoritmo recursivo que percorre toda a **árvore** (fluxograma) em profundidade e copia o conteúdo da propriedade *text* (linhas de texto) de cada objeto do componente *ExpressFlowChart* (DEVELOPER, 2004), para o componente *memo* do Delphi. Enquanto o algoritmo realiza a busca recursiva no fluxograma, são realizadas diversas consistências entre os objetos e conexões criados. Entre as consistências que são realizadas pode-se citar a verificação das ligações dos blocos de *goto* e a verificação de blocos que indicam o final de um fluxo.

```
{ Criar o Objeto Processo no fluxograma. }
procedure CriarProcesso(aph,apv,ath,atv: integer);
var
    xtriangulo: Boolean;
begin
   xtriangulo := False;
   { Verifica qual objeto do fluxograma está selecionado. }
   if gNomeFlow.SelectedObject.ShapeType <> fcsSouthTriangle then
   begin
        { Chama rotina que cria um objeto (LABEL) na forma de triangulo de cor branca. }
        CriaLable(aph,apv,ath,atv);
        FrmPrincipal.SetaMedidasObjeto;
   End7
   Else
        xtriangulo := true;

   { Verifica o quantidade de objetos que já foram criados nos fluxogramas. }
   oind := gNomeFlow.ObjectCount + gNomeOutroFlow.ObjectCount;
   { Cria um novo objeto no fluxograma com a forma de um retângulo e atribui algumas propriedades. }
   oObjeto[oind] := gNomeFlow.CreateObject(aph, apv, ath,atv, fcsRectangle);
   oObjeto[oind].CustomData := 'PRO' + IntToStr(oind);
   oObjeto[oind].HorzTextPos := fchpLeft;
   oObjeto[oind].BkColor := $007BB9F0;

   if not xtriangulo then
   begin
      { Cria uma conexão entre o objeto que foi selecionado no fluxograma e o label criado. }
      cConexao[cind] := gNomeFlow.CreateConnection(gNomeFlow.Objects
                          [gNomeFlow.SelectedObject.ZOrder],oObjeto[oind-1],10,2);

      { Chama rotina que define o sentido da flecha na conexão. }
      ConfiguraFlechaConexao;
      Inc(cind);
      { Cria uma conexão entre o label criado e o objeto criado. }
      cConexao[cind] := gNomeFlow.CreateConnection(oObjeto[oind-1],oObjeto[oind],10,2);
      ConfiguraFlechaConexao;
      Inc(cind);
   End
   Else
   Begin
      { Cria uma conexão entre o objeto que foi selecionado no fluxograma e o objeto criado.}
      Cconexao[cind] := gNomeFlow.CreateConnection(gNomeFlow.Objects
                          [gNomeFlow.SelectedObject.ZOrder],oObjeto[oind],10,2);
      ConfiguraFlechaConexao;
      Inc(cind);
   End;
End;
```

Quadro 4 – Exemplo do código fonte para criar um objeto e uma conexão no fluxograma

Um trecho do código fonte que mostra o uso do algoritmo recursivo pode ser visto no Quadro 5.

```
{ Verifica o Nó Fluxograma para gerar o código fonte. }
procedure TFrmPrincipal.VerificaNo(fluxograma: TdxFlowChart; xno: String);
var   x: Integer;
begin
   { Varre todas as conexoes verificando o conteúdo dos objetos no fluxograma. }
   for x := 0 to fluxograma.ConnectionCount - 1 do
   begin
      // Tratamento especial para GOTO.
      If (fluxograma.Connections[x].ObjectSource.CustomData = xno) and
         (fluxograma.Connections[x].PenStyle <> psDash) and (not gFalhou) then
      begin
         { Verifica se todos os blocos finais estão criados. }
         if (fluxograma.Connections[x].ObjectDest.ConnectionCount = 1) and
            ((fluxograma.Connections[x].ObjectDest.ShapeType = fcsSouthTriangle) or
            (fluxograma.Connections[x].ObjectDest.ShapeType = fcsRectangle)) then
         begin
            gFalhou := True;
            if (fluxograma = dxFlowChart1) then
            begin
               gNomeFlow := dxFlowChart1;
               gNomeOutroFlow := dxFlowChart0;
               SPBCompilar.Enabled := False;
               Menu.Items[3].Items[5].Enabled := False;
               PageControl.ActivePageIndex := 2;
            end;
            fluxograma.ClearSelection; // Tira a seleção de todos os objetos.
            fluxograma.Connections[x].ObjectDest.Selected := True;
            ShowMessage('Não foi Possível Compilar o Fluxograma. O Objeto Selecionado não Possui uma
                        Conexão de FIM.');
            Fluxograma.SetFocus;
            Break;
         End;
         { Só adiciona o texto ao Memo se o objeto possuir algum texto. }
         if (fluxograma.Connections[x].ObjectDest.Text <> '') then
            MemoFonte.Lines.Text := MemoFonte.Lines.Text + "#13" +
            (fluxograma.Connections[x].ObjectDest.Text);
         VerificaNo(fluxograma, fluxograma.Connections[x].ObjectDest.CustomData);
      End;
   End;
End;

Procedure TFrmPrincipal.Compilar1Click(Sender: TObject);
Var
  Xfluxograma : TdxFlowChart;
  i : integer;
begin
  { DIVERSAS CONSISTENCIAS .....}

  { ********** FLUXOGRAMA PRINCIPAL ********** }
  MemoFonte.Lines.Text := MemoFonte.Lines.Text + "#13" + (xfluxograma.Objects[0].text);
  VerificaNo(xfluxograma, xfluxograma.Objects[0].CustomData);

  { DIVERSAS CONSISTENCIAS .....}
End;
```

Quadro 5 – Trechos do algoritmo de compilação dos fluxogramas

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Esta seção descreve a funcionalidade do protótipo através de um estudo de caso que simula um alarme residencial.

3.3.2.1 Circuito para testes

Para realizar os testes na implementação, é utilizado o circuito visto na Figura 15, que consiste de sensores (representados por chaves) e sinalizadores (representados por *led's*).

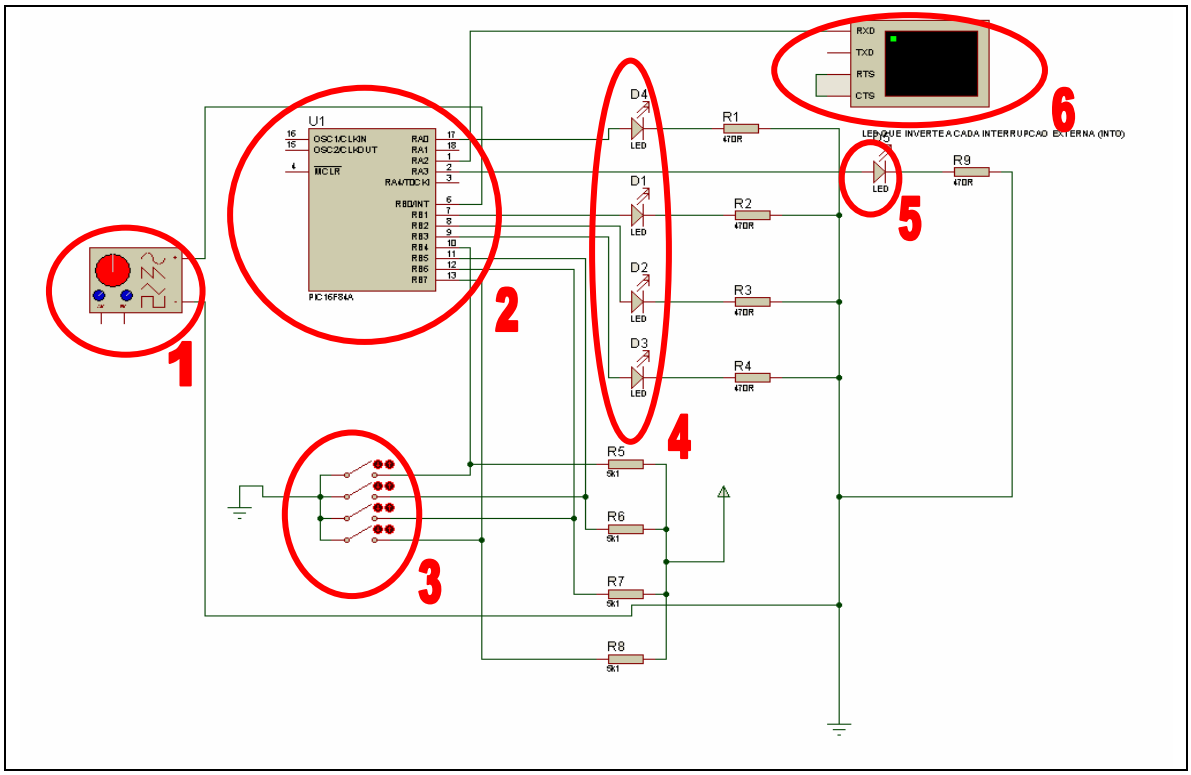


Figura 15 – Circuíto de testes utilizando o Proteus

Na Figura 15 observa-se que circuito é composto pelos seguintes componentes indicados pelos círculos:

a) círculo 1: um gerador de sinal que possui a função de gerar um pulso (onda quadrada) a cada um segundo para forçar uma interrupção externa que possui a função de indicar que o circuito está funcionando;
b) círculo 2: um microcontrolador PIC16F84A que executa o programa gerado pelo protótipo que monitora os sensores;
c) círculo 3: quatro sensores que simulam os pontos monitoração;

d) círculo 4: quatro sinalizadores que possuem a função de indicar qual sensor está atuando;

e) círculo 5: um sinalizador que possui a função de indicar quando ocorre uma interrupção externa;

g) círculo 6: terminal que possui a mesma função do sinalizador.

A seguir estão descritas as principais telas do protótipo simulando o circuito demonstrado na Figura 15.

3.3.2.2 Declarações e Definições

A Figura 16 demonstra a tela principal do protótipo, que é composta por uma barra de menus que provê o acesso a todas as funções disponíveis no ambiente, uma barra de botões de acesso rápido, que é composta pelas opções do menu mais utilizadas durante o processo de construção de um fluxograma e três abas, sendo "Fluxograma", "Código Fonte" e "Interrupção", respectivamente. Esta última, somente torna-se visível quando o usuário gera um bloco de interrupção no fluxograma principal.

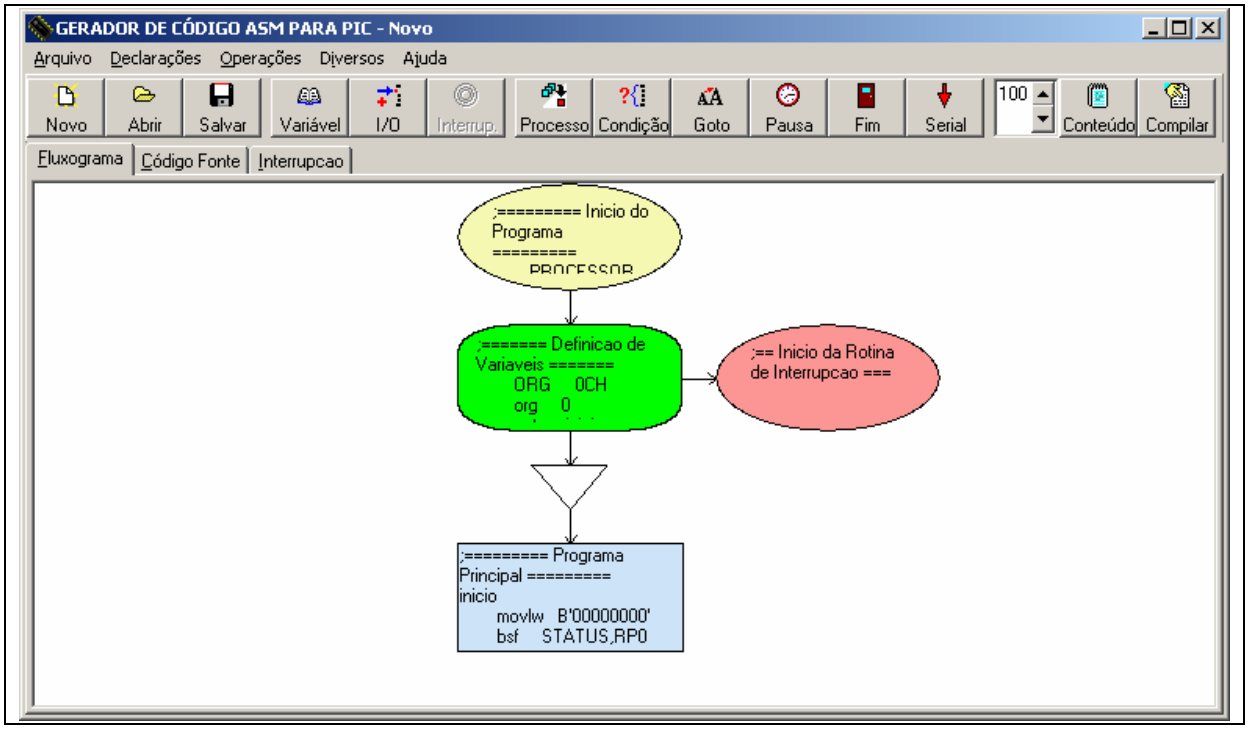


Figura 16 – Tela principal do protótipo

No menu “Arquivo”, estão dispostas algumas opções que possuem funcionalidades semelhantes à de outros aplicativos Microsoft como, por exemplo, o Microsoft Word. Ele é composto pelos seguintes submenus: “Novo”, “Abrir”, “Fechar”, “Salvar”, “Salvar Como...”, “Imprimir Código Fonte” e “Sair”, respectivamente.

O menu “Declarações” possui três sub-menus denominados de “Variáveis”, “I/O” e “Interrupção”, respectivamente. Ele é responsável pelo processo de declarações de variáveis, onde são realizadas diversas consistências como, por exemplo, verificar se a variável que está sendo inserida possui o mesmo nome de um registrador do PIC16F84A, pelo processo de definições de entradas/saídas do microcontrolador e por criar o bloco de interrupção no fluxograma principal.

Para o usuário criar um fluxograma a fim de simular o estudo de caso apresentado anteriormente, primeiro deve pressionar o botão “Novo” para criar um novo fluxograma. Em seguida, o usuário deve declarar as variáveis que são usadas no programa e definir as configurações de entradas/saídas clicando no botão “I/O”. A tela de configurações de entradas/saídas pode ser vista na Figura 17.

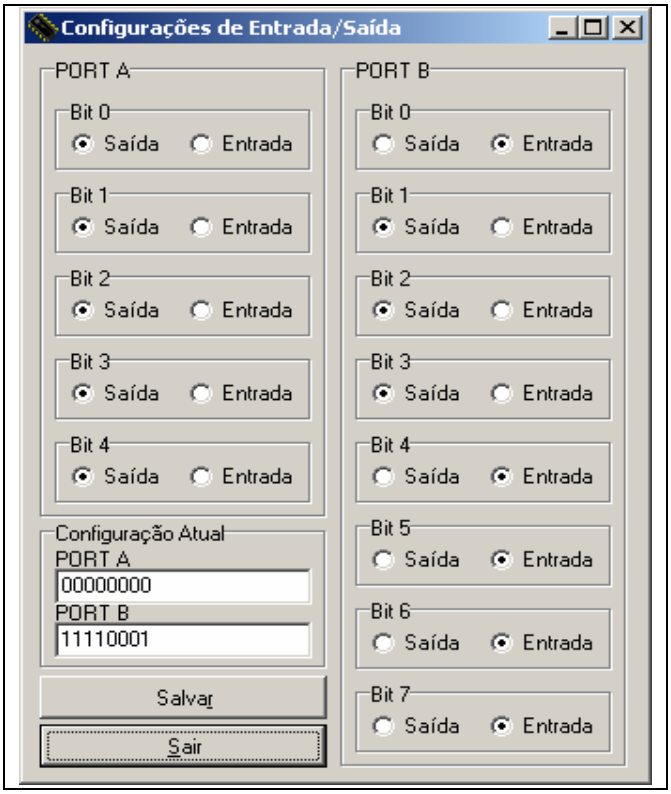


Figura 17 – Tela de configurações de entrada/saída

Na tela apresentada pela Figura 17, o usuário define os *bits* *PORT*A e *PORT*B do microcontrolador, que serão utilizados como entrada ou saída durante a execução do programa. O botão "Salvar" grava as alterações realizadas pelo usuário e atualiza os campos *PORT* A e *PORT*B no item "Configuração Atual" da tela, gerando a representação das portas em código binário. O botão "Sair" fecha a tela e atualiza o nodo "Início" (representado por um retângulo) no fluxograma principal. Se o usuário realizar alterações nesta tela e não clicar no botão "Salvar", as novas configurações serão ignoradas.

3.3.2.3 Interrupções

Após realizar as configurações descritas acima, o usuário deve pressionar o botão "Interrup" para gerar o bloco de interrupção no fluxograma principal. Ao gerar este bloco é criada uma aba ao lado direito da aba "Código Fonte" no protótipo, para edição e construção de um fluxograma específico para o tratamento das interrupções.

No fluxograma principal é criado um objeto que recebe o nome de "Início da rotina de interrupção" (representado por uma elipse), que sinaliza ao fluxograma principal a existência de um fluxograma específico para o tratamento das interrupções. A criação do bloco de interrupção pode ser visto na Figura 18.

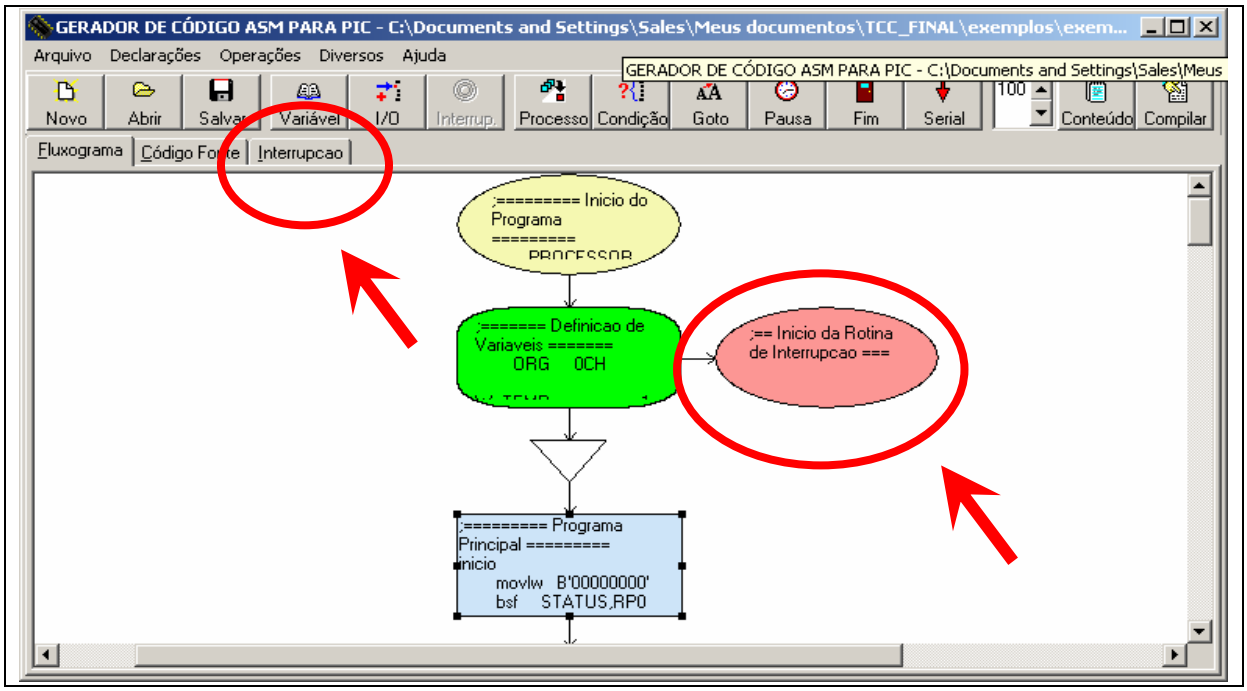


Figura 18 – Criação do bloco e da aba de Interrupção

Ao criar este bloco, o sistema adiciona ao bloco de declaração de variáveis um desvio para o tratador de interrupção através do texto “org 4”, que representa o endereço de memória no qual o programa será desviado ao ocorrer uma interrupção e o texto “goto int_prin”, que realiza o desvio para a rotina de interrupção, como visto na Figura 19.

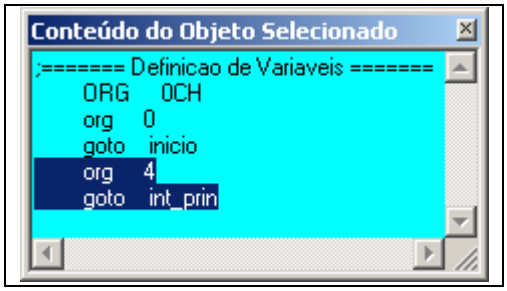


Figura 19 – Texto da interrupção no bloco de declaração de variáveis

A habilitação das interrupções fica a critério do usuário e deve ser realizada no fluxograma principal, através dos blocos de processo que serão descritos adiante. Todos os tratamentos das interrupções habilitadas devem ser realizados no fluxograma de interrupção.

Para este estudo de caso, o usuário deve habilitar a interrupção global e a interrupção externa no fluxograma principal, conforme visto na Figura 20.

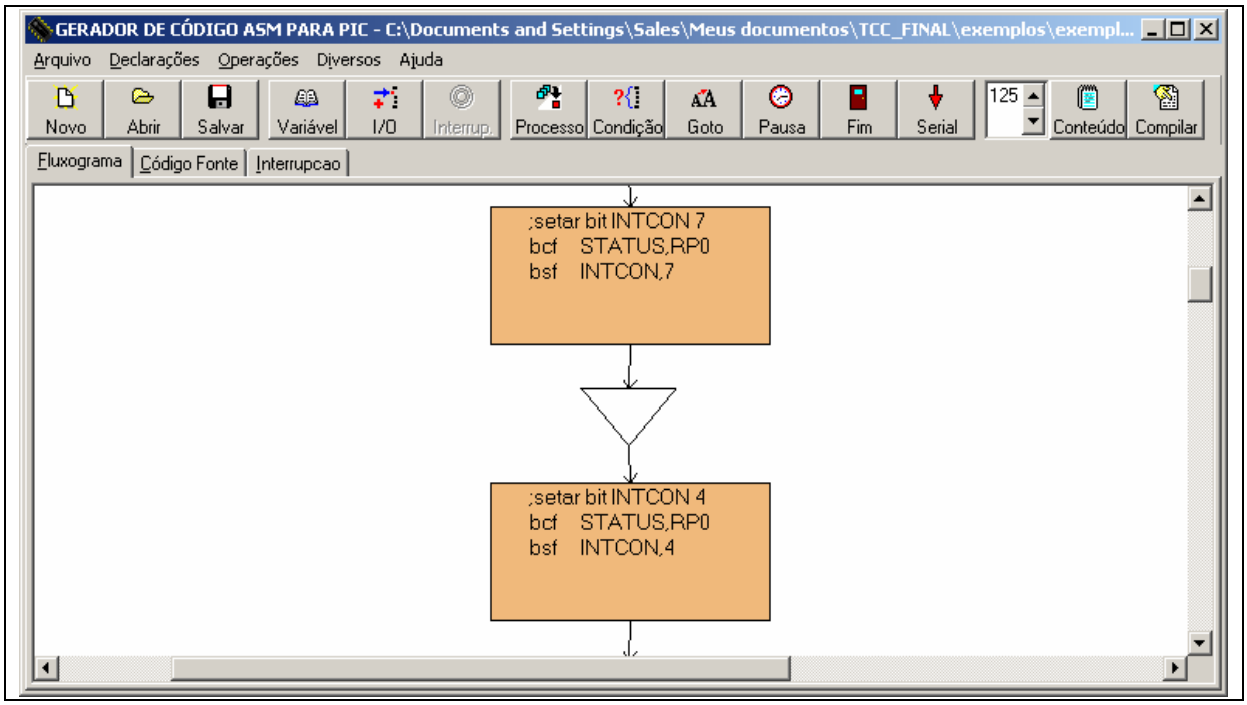


Figura 20 – Habilitação das interrupções global e externa

O fluxograma de interrupção que realiza o tratamento de todas as interrupções habilitadas pode ser visto na Figura 21.

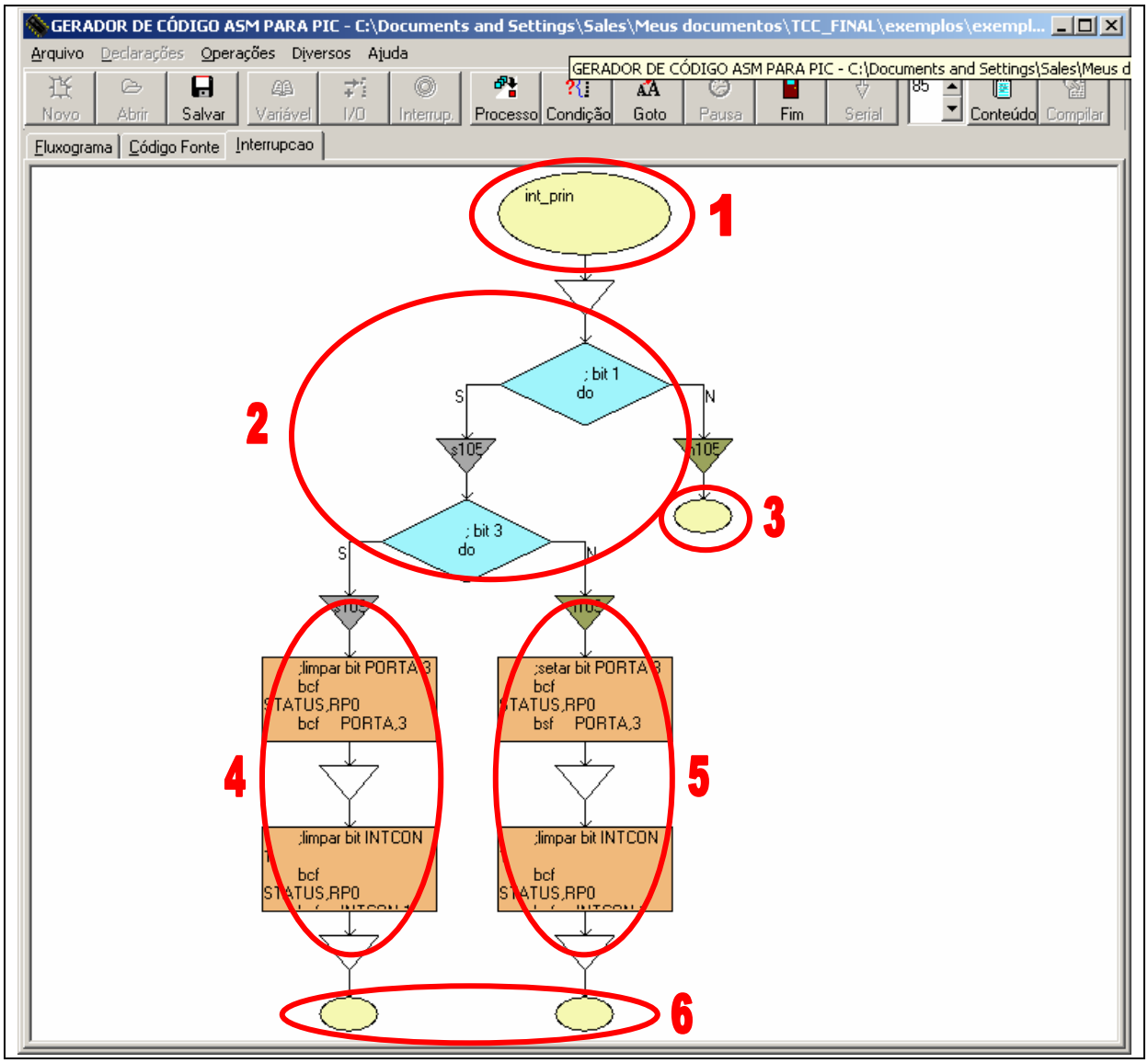


Figura 21 – Fluxograma de interrupção

Este fluxograma da Figura 21 está estruturado com os seguintes blocos:

a) círculo 1: bloco de início do fluxograma de interrupção;

b) círculo 2: dois blocos de condição, sendo que o bloco que contém o texto “bit1” (superior) verifica se o *bit* da habilitação global das interrupções está habilitado e, caso não esteja, vai para o círculo 3; e o bloco que contém o texto “bit3” (inferior) verifica se a interrupção externa está habilitada e, se estiver, vai para o círculo 4, senão, vai para o círculo 5;

c) círculo 3: um bloco que indica o término da rotina de interrupção;
d) círculo 4: dois blocos de processo, sendo que o superior é responsável por apagar o *led* que está aceso no circuito e o bloco inferior desabilita as interrupções externas;
e) círculo 5: dois blocos de processo, sendo que o superior é responsável por acender o *led* que está apagado no circuito e o bloco inferior desabilita as interrupções externas;
f) círculo 6: dois blocos que indicam o término da rotina de interrupção;

Após o término da execução do fluxograma de interrupção, o programa retorna a sua execução normal a partir do ponto onde parou.

3.3.2.4 Processos

O menu Processo é responsável por criar um bloco de processo no fluxograma ativo. Para criar este tipo bloco é necessário selecionar um objeto do fluxograma que possua uma única conexão ligada a ele e deve-se pressionar o botão "Processo" na barra de botões. Durante a abertura da tela indicada na Figura 22, o protótipo realiza uma série de consistências relacionadas a conexões entre os objetos para impedir que o usuário crie objetos que possam gerar erros ao compilar os fluxogramas.

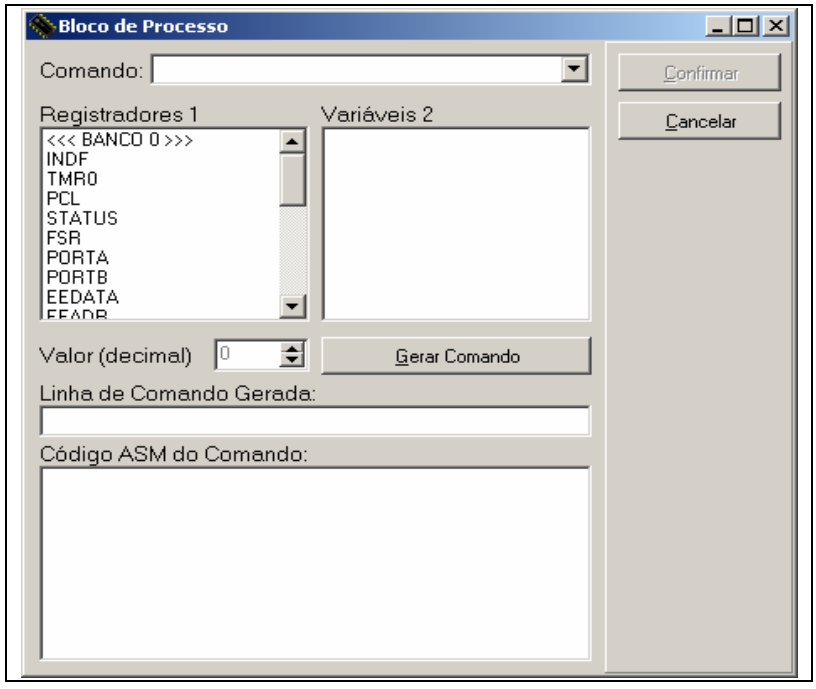


Figura 22 – Tela de processos

No *combobox* “Comando” localizado na parte superior da tela, tem-se diversas opções para manipulação de registradores, variáveis e literais conforme indicado na Tabela 3.

Tabela 3 – Opções do *combobox* “Comando” na tela de processos

Opção Comando	Código Gerado em Assembly
decrementar registrador	bcf STATUS,RP0 decf PORTA,f { decrementa o registrador PORTA }
decrementar variavel	decf variavel,f { decrementa 1 da variável 1 }
incrementar registrador	bcf STATUS,RP0 incf PORTA,f { incrementa o registrador PORTA }
incrementar variavel	incf variavel,f { incrementa 1 a variável 1 }
limpar bit registrador	bcf STATUS,RP0 bcf PORTA,valor { limpar o bit valor do registrador PORTA }
limpar bit variavel	bcf variavel1,valor { limpa o bit valor da variável 1 }
mover literal para registrador	bcf STATUS,RP0 movlw valor movwf PORTA { move o literal valor para o registrador PORTA }
mover literal para variavel	movlw valor movwf variavel1 { mover o valor para a variável 1 }
mover literal para w	movlw valor { move o valor para w }
mover registrador para registrador	bcf STATUS,RP0 movf PORTA,w bcf STATUS,RP0 movwf PORTB { move o registrador PORTA para o registrador PORTB setando o banco }
mover registrador para variavel	bcf STATUS,RP0 movf PORTB,w movwf variavel1 { move o registrador PORTB para a variavel variavel 1 }
mover registrador para w	bcf STATUS,RP0 movf PORTB,w { move o registrador PORTB para w }
mover variavel para registrador	bcf STATUS,RP0 movf variavel1,w movwf PORTB { move o valor da variavel 1 para o registrador PORTB }
mover variavel para variavel	movf variavel1,w movwf variavel2 { move o valor da variável 1 para a variável 2 }
mover variavel para w	movf variavel1,w { move a variável 1 para w }
setar bit registrador	bcf STATUS,RP0 bsf PORTA,valor { seta o bit valor do registrador PORTA }
setar bit variavel	bsf variavel1,valor { seta o bit valor da variável 1 }

As variáveis que são apresentadas na lista de variáveis desta tela são carregadas de uma lista definida na tela de declaração de variáveis. A lista de registradores é uma lista fixa e não pode ser alterada pelo usuário. As opções das listas de variáveis e/ou registradores tornam-se visíveis ao usuário de acordo com a opção definida no *combobox* "Comando".

Após o usuário definir qual o comando e qual o registrador e/ou variável e/ou literal deve fazer parte do bloco de processo, o mesmo deve pressionar no botão "Gerar Comando" que realiza algumas consistências entre os campos escolhidos pelo usuário e possui a funcionalidade de gerar o código em Assembly a partir das opções definidas na tela.

No campo "Linha de Comando Gerada" é apresentada uma explicação do comando que é gerado. Já no *memo* "Código ASM do Comando" é gerado o código em Assembly que é incluso no bloco de processo do fluxograma quando o usuário pressiona o botão "Confirmar". Neste *memo,* o usuário pode alterar o código que é gerado pelo botão "Gerar Comando" antes de criar o bloco de processo no fluxograma. Esta opção de editar o código gerado é muito útil quando o usuário necessita criar um bloco de processo em que o protótipo não contemple as opções desejadas pelo usuário.

Para o estudo de caso, o usuário deve criar diversos blocos de processo. Um exemplo pode ser visto na Figura 23.

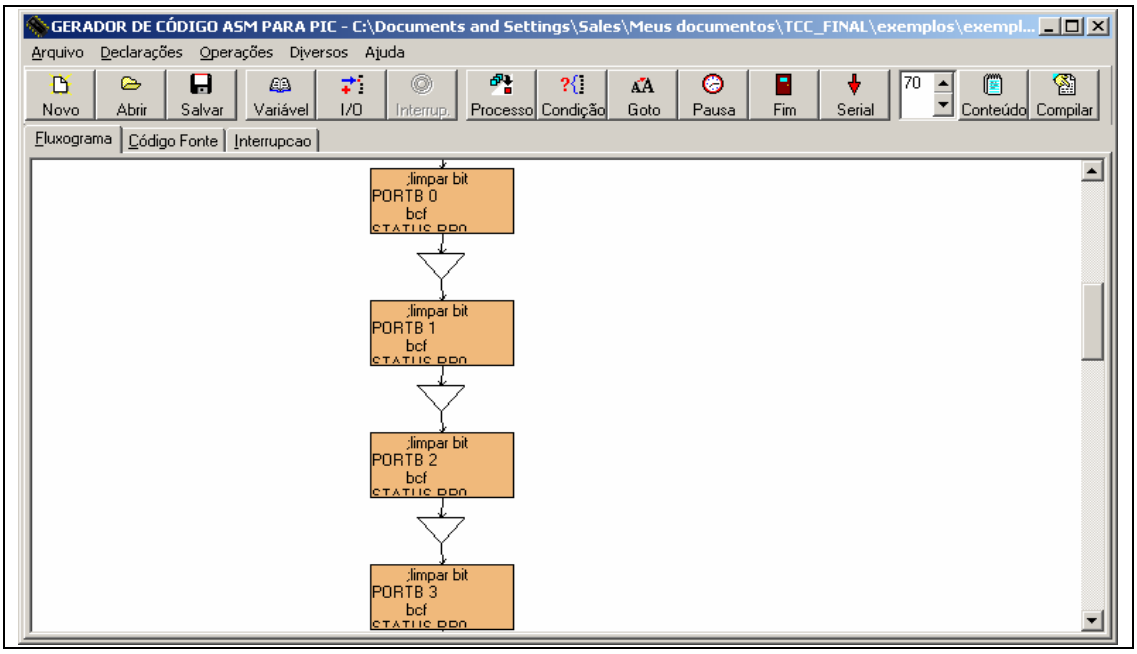


Figura 23 – Exemplos de blocos de processos

A seqüência de blocos da Figura 23 representa que o usuário deve forçar o programa a apagar todos os *led's* dos *bits* 0, 1, 2 e 3 do *PORTB* .

3.3.2.5 Condição

O menu "Condição" é responsável por criar um bloco de condição no fluxograma ativo. O princípio de funcionamento desta tela segue o mesmo padrão definido para os blocos de processo, diferenciando-se em algumas consistências internas e por possuir o campo "Condição" com alguns operadores lógicos (>, < e =) de condição.

A tela de condições pode ser vista na Figura 24.

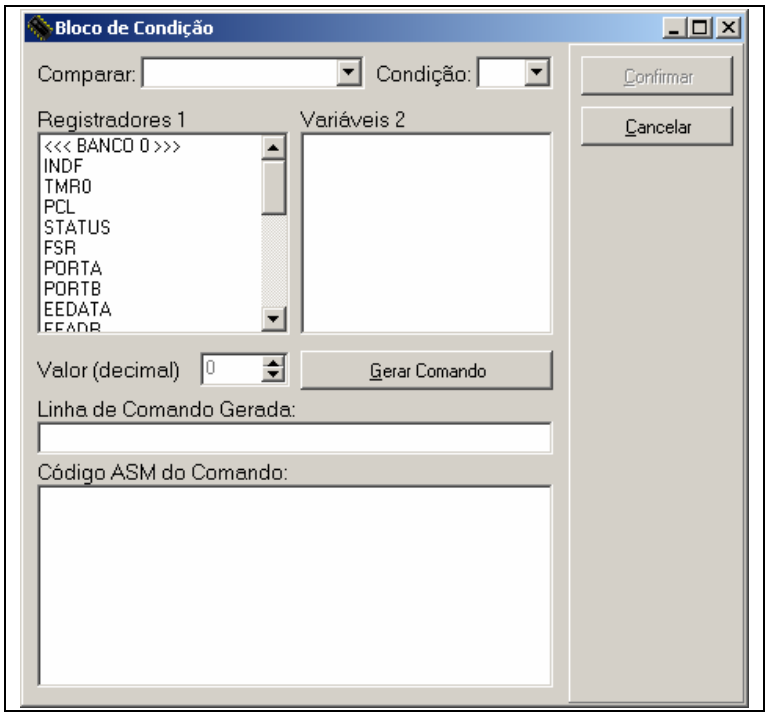


Figura 24 – Tela de condições

No *combobox* "Comparar", localizado na parte superior da tela, tem-se diversas opções para manipulação de registradores, variáveis e literais conforme descrito na Tabela 4.

Tabela 4 – Opções do *combobox* "Comparar" na tela de condições

Opção Comparar	Código Gerado em Assembly
bit registrador = clear	bcf STATUS,RP0 btfsc PORTB,valor { bit valor do registrador PORTB = clear! }
bit registrador = set	bcf STATUS,RP0 btfss PORTB,valor { bit valor do registrador PORTB = set! }
bit variável = clear	btfsc variavel1,valor { bit valor da variável 1 = clear! }
bit variável = set	btfss variavel1,valor { bit valor da variável 1 = set! }
registrador com literal	bcf STATUS,RP0 movlw valor subwf PORTA * operador do comando { PORTA com valor }
registrador com registrador	bcf STATUS,RP0 movf PORTA,w bcf STATUS,RP0 subwf PORTB * operador do comando { PORTA com PORTB }
registrador com variavel	bcf STATUS,RP0 movf variavel1,w subwf PORTA * operador do comando { PORTA com variavel 1 }
variavel com literal	movlw valor subwf variavel1,w * operador do comando { variável 1 com valor}

variavel com variavel	movf variavel2,w
	subwf variavel1,w
	* operador do comando
	{ variável 1 com variável 2 }

* operador do comando:
se o operador for o sinal de igual (=): comando = btfss STATUS,z,
se o operador for o sinal de maior (>): comando = btfsc STATUS,c
se o operador for o sinal de menor (<): comando = btfss STATUS,c

STATUS é um registrador do PIC16F84A que, segundo Souza (2002, p. 17) é "utilizado para mostrar o estado da unidade aritmética e lógica (ULA), a forma do último *reset* e também para configurar a página de programação atual, quando necessário".

No estudo de caso, o usuário faz uso dos blocos de condição para verificar qual é a chave que está sendo pressionada no circuito. Estes blocos são vistos na Figura 25.

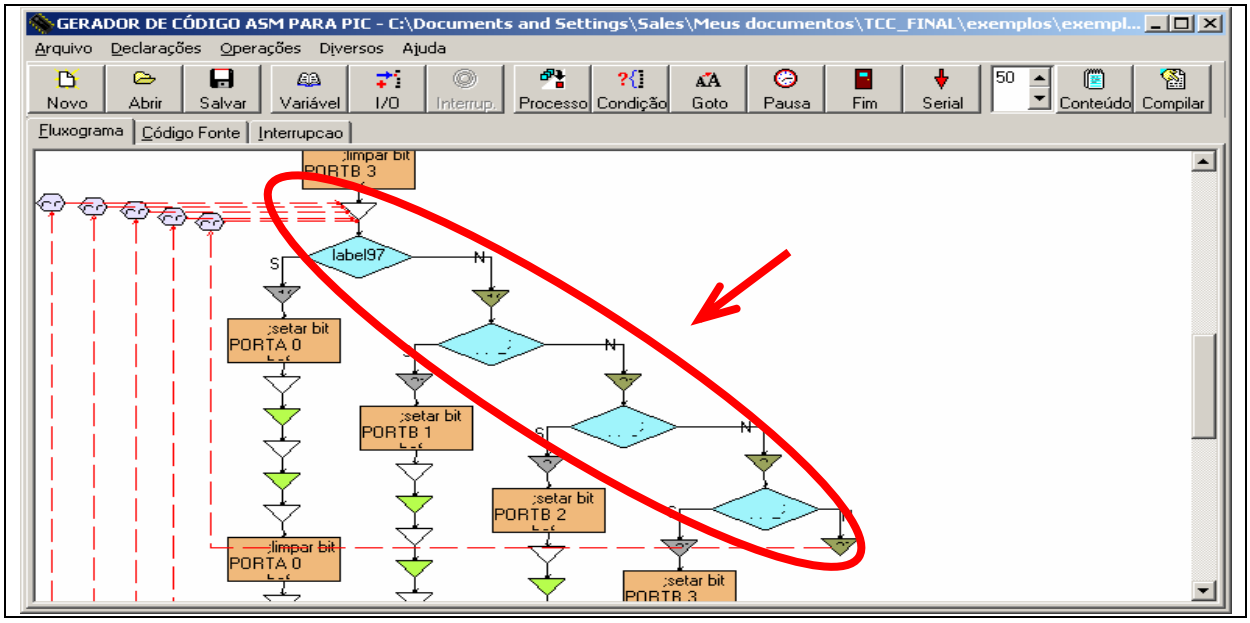


Figura 25 – Blocos do processo do estudo de caso

3.3.2.6 Goto

O “goto” no fluxograma é utilizado para gerar um desvio no fluxo normal do programa. Para criar um bloco de goto, o usuário deve selecionar um bloco de origem, clicar no botão goto, selecionar um bloco de destino e clicar no botão goto novamente. Seguindo este procedimento, uma conexão com um bloco representado por um hexágono é criada no fluxograma ativo e são criadas as conexões entre os objetos origem e *goto* e os objetos *goto* e destino. Estas conexões são representadas por linhas tracejadas na cor vermelha.

A Figura 26 mostra o uso de blocos de goto no fluxograma.

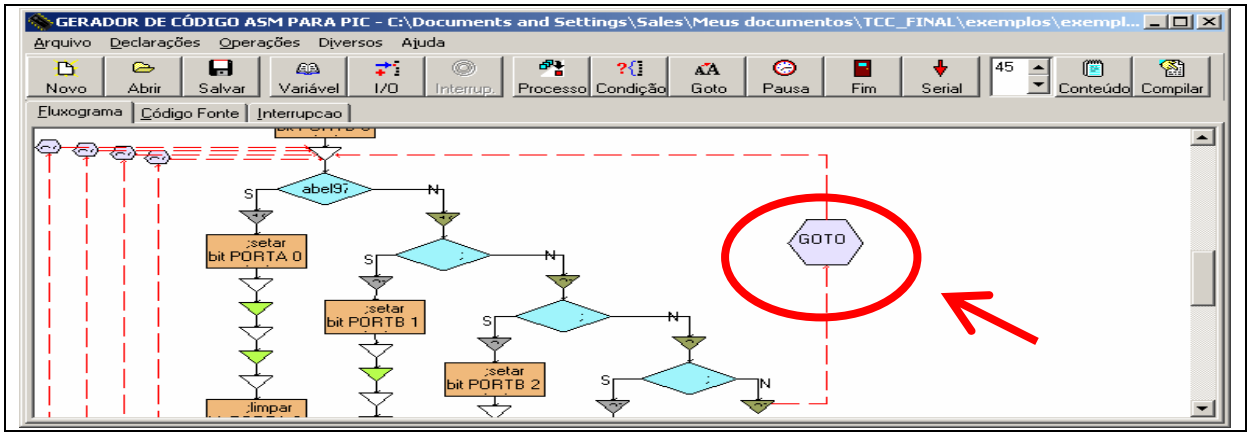


Figura 26 – Bloco *goto* do estudo de caso

3.3.2.7 Pausa

O menu “Pausa” é responsável por criar um bloco de pausa no fluxograma ativo. Quando este bloco é criado pela primeira vez é carregada uma tela, conforme visto na Figura 27, que solicita ao usuário que informe a quantidade de tempo, em milissegundos, que cada bloco permanecerá executando. Este tempo, definido pelo usuário pode variar de 0 a 255 milissegundos (baseado em um cristal de 4MHZ). Por exemplo, se o usuário deseja criar uma pausa de um segundo, ele deve criar quatro blocos de pausa de 255 milisegundos seguidos no fluxograma. Este menu somente está disponível para o fluxograma principal.

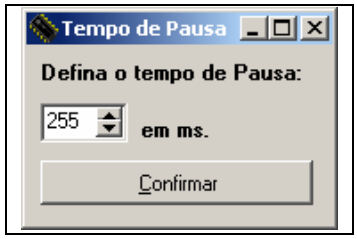


Figura 27 – Tela de pausa

A representação dos blocos de pausa no fluxograma do estudo de caso pode ser vista na Figura 28, onde a seqüência de blocos inseridos dentro do círculo representa uma pausa de meio segundo.

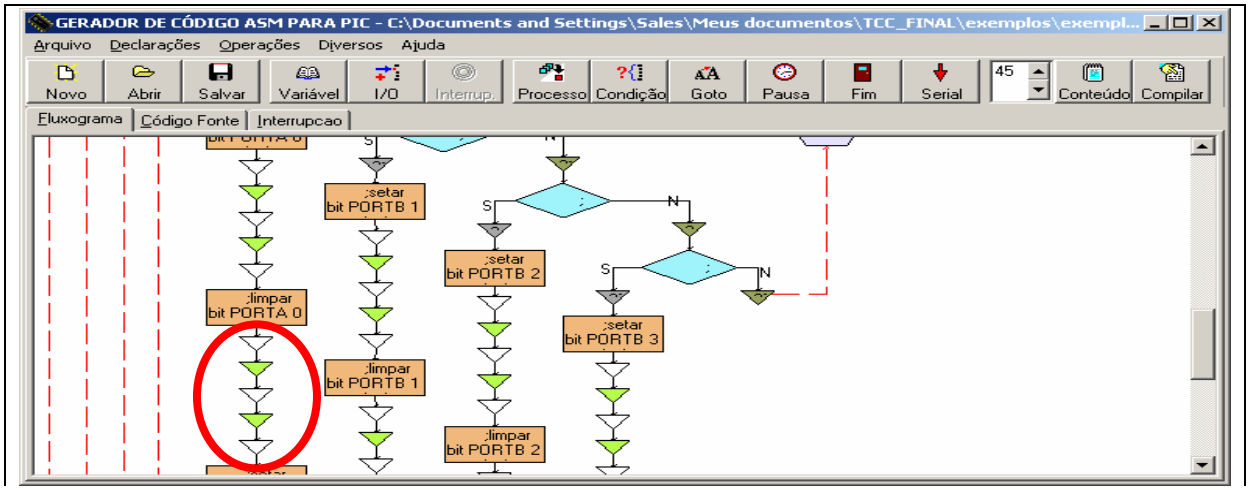


Figura 28 – Blocos de pausa no estudo de caso

3.3.2.7.1 Fim e Serial

O bloco “Fim” tem a função de criar um objeto com o texto “goto fim” que realiza um desvio para o final do programa e é representado por uma elipse amarela.

O bloco “Serial” demonstrado através da Figura 29 insere um objeto representado por um triângulo de cor vermelha no fluxograma. O objeto criado possui o texto “call TxChar” que realiza um desvio para a rotina que imprime um caractere serialmente em uma porta do PIC. A porta responsável por imprimir na serial é definida pelo usuário ao compilar o fluxograma. Esta porta deve estar configurada para transmitir a uma taxa de 19.200 *baud,* 8 *databits*, sem paridade e 1 *stopbit*. Esta opção está disponível somente para o fluxograma principal.

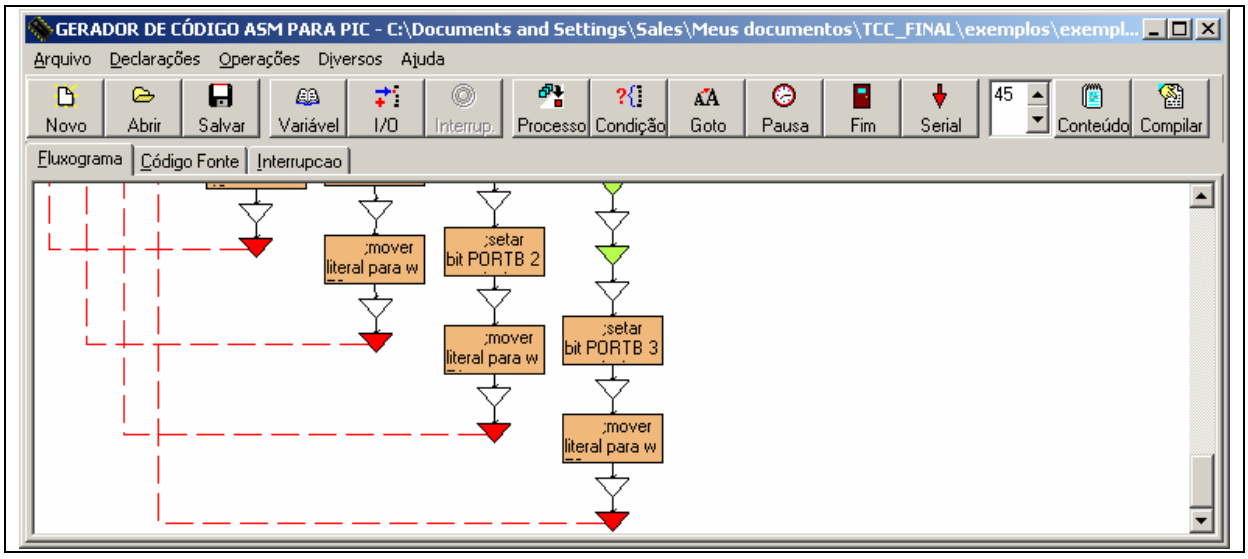


Figura 29 – Bloco serial no estudo de caso

3.3.2.8 Demais funcionalidades do protótipo

O menu “Diversos” possui algumas opções para o usuário trabalhar com o fluxograma. Neste menu existem as opções de “Selecionar Tudo” e “Desfazer Seleção”, têm a função de selecionar e desfazer a seleção de todos os objetos do fluxograma, respectivamente. Apenas os objetos são selecionados ou desselecionados. As conexões não são selecionadas junto com os objetos. Estas funções são úteis quando o usuário necessita mudar a posição do fluxograma dentro da área de edição.

As opções “Excluir Objeto Selecionado” e “Excluir Conexão Selecionada” possuem a função de excluir um objeto e uma conexão previamente selecionada, respectivamente. Somente é possível excluir um objeto ou uma conexão por vez.

Ainda neste menu existe a opção para o usuário carregar uma tela de conteúdo que tem o objetivo de mostrar o conteúdo de um objeto selecionado no fluxograma. Para o usuário visualizar o texto contido em um objeto previamente criado sem necessitar redimensioná-lo, deve-se abrir esta tela. Quando houver um objeto selecionado, o conteúdo do mesmo será mostrado nesta tela mas não pode ser editado, pois a função desta tela é única e exclusivamente para visualizar o texto do objeto selecionado.

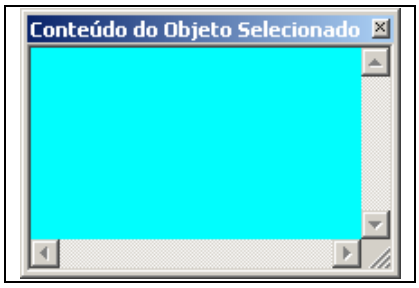


Figura 30 – Tela de Conteúdo

A função “Carregar *MPLAB*” possibilita ao usuário abrir o aplicativo *MPLAB* (MICROCHIP, 2004) quando o mesmo estiver instalado no diretório padrão de instalação na máquina onde o protótipo estiver rodando.

3.3.2.9 Compilar

Através desta opção é realizada a compilação do fluxograma. Quando o usuário clica nesta opção, o algoritmo de compilação realiza uma busca recursiva em todos os objetos do fluxograma principal e posteriormente no fluxograma de interrupção, se o mesmo foi criado. Durante o processo de busca, o algoritmo realiza diversas consistências, entre as quais verifica se todos os objetos do fluxograma possuem conexões que possuam algum objeto ligado.

3.4 RESULTADOS E DISCUSSÕES

Para validar o protótipo foram realizados testes simulando o estudo de caso mencionado anteriormente. Através deste estudo de caso, pode-se verificar a eficácia do

protótipo em gerar código Assembly, conforme visto no Quadro 6 para o microcontrolador PIC16F84A.

Através do circuito apresentado, foi demonstrado o funcionamento das principais rotinas disponíveis no protótipo, entre as quais pode-se destacar o uso das rotinas de interrupção, de pausa e de escrita em serial.

```
;========= Inicio do Programa =========
      PROCESSOR       PIC16F84A
      INCLUDE "P16F84A.INC"
      RADIX   DEC
      TX            equ     2
      BIT_DELAY      equ     23
;======= Definicao de Variaveis =======
      ORG     0CH
W_TEMP          res     1
STATUS_TEMP     res     1
pausat          res     1
pausad          res     1
msDelayCounter  res     2
TmpRegister     res     1
ShiftReg        res     1
BitCount        res     1
      org     0
      goto    inicio
      org     4
      goto    int_prin
;========= Programa Principal =========
inicio
      movlw   B'00000000'
      bsf     STATUS,RP0
      movwf   TRISA
      movlw   B'11110001'
      movwf   TRISB
      bcf     STATUS,RP0
      ;setar bit INTCON 7
      bcf     STATUS,RP0
      bsf     INTCON,7
      ;setar bit INTCON 4
      bcf     STATUS,RP0
      bsf     INTCON,4
      ;limpar bit PORTB 0
      bcf     STATUS,RP0
      bcf     PORTB,0
;limpar bit PORTB 1
      bcf     STATUS,RP0
      bcf     PORTB,1
      ;limpar bit PORTB 2
      bcf     STATUS,RP0
      bcf     PORTB,2
      ;limpar bit PORTB 3
      bcf     STATUS,RP0
      bcf     PORTB,3
      goto    label97
label97
      ; bit 4 do registradorl PORTB = clear!
      bcf     STATUS,RP0
      btfsc   PORTB,4
      goto    n18
      goto    s18
```

```
s18
      ;setar bit PORTA 0
      bcf     STATUS,RP0
      bsf     PORTA,0
      call    pausap
      call    pausap
      ;limpar bit PORTA 0
      bcf     STATUS,RP0
      bcf     PORTA,0
      call    pausap
      call    pausap
      ;setar bit PORTA 0
      bcf     STATUS,RP0
      bsf     PORTA,0
      ;mover literal para w 49
      movlw   49
      call    TxChar
      goto    label97
n18
      ; bit 5 do registradorl PORTB = clear!
      bcf     STATUS,RP0
      btfsc   PORTB,5
      goto    n21
      goto    s21
s21
      ;setar bit PORTB 1
      bcf     STATUS,RP0
      bsf     PORTB,1
      call    pausap
      call    pausap
      ;limpar bit PORTB 1
      bcf     STATUS,RP0
      bcf     PORTB,1
      call    pausap
      call    pausap
      ;setar bit PORTB 1
      bcf     STATUS,RP0
      bsf     PORTB,1
      ;mover literal para w 50
      movlw   50
      call    TxChar
      goto    label97
n21
      ; bit 6 do registradorl PORTB = clear!
      bcf     STATUS,RP0
      btfsc   PORTB,6
      goto    n24
      goto    s24
```

```
s24
      ;setar bit PORTB 2
      bcf    STATUS,RP0
      bsf    PORTB,2
      call   pausap
      call   pausap
      ;limpar bit PORTB 2
      bcf    STATUS,RP0
      bcf    PORTB,2
      call   pausap
      call   pausap
      ;setar bit PORTB 2
      bcf    STATUS,RP0
      bsf    PORTB,2
      ;mover literal para w 51
      movlw  51
      call   TxChar
      goto   label97
n24
      ; bit 7 do registradorl PORTB = clear!
      bcf    STATUS,RP0
      btfsc  PORTB,7
      goto   n27
      goto   s27
s27
      ;setar bit PORTB 3
      bcf    STATUS,RP0
      bsf    PORTB,3
      call   pausap
      call   pausap
      ;limpar bit PORTB 3
      bcf    STATUS,RP0
      bcf    PORTB,3
      call   pausap
      call   pausap
      ;setar bit PORTB 3
      bcf    STATUS,RP0
      bsf    PORTB,3
      ;mover literal para w 52
      movlw  52
      call   TxChar
      goto   label97
n27
      goto   label97
;== Inicio da Rotina de Interrupcao ===
int_prin
      ;Salava o valor de W e STATUS
      MOVWF  W_TEMP     ;salva W em W_TEMP
      SWAPF  STATUS,W
      MOVWF  STATUS_
      ; bit 1 do registradorl INTCON = set!
      bcf    STATUS,RP0
      btfss  INTCON,1
      goto   n105
      goto   s105
s105
      ; bit 3 do registradorl PORTA = set!
      bcf    STATUS,RP0
      btfss  PORTA,3
      goto   n109
      goto   s109
s109
      ;limpar bit PORTA 3
      bcf    STATUS,RP0
      bcf    PORTA,3
      ;limpar bit INTCON 1
      bcf    STATUS,RP0
      bcf    INTCON,1
      goto   fim_int
n109
      ;setar bit PORTA 3
      bcf    STATUS,RP0
      bsf    PORTA,3
      ;limpar bit INTCON 1
      bcf    STATUS,RP0
      bcf    INTCON,1
      goto   fim_int
n105
      goto   fim_int
fim_int
      ;Recupera o valor de W e STATUS
      SWAPF  STATUS_TEMP,W
      MOVWF  STATUS   ;recupera STATUS
      SWAPF  W_TEMP,F
      SWAPF  W_TEMP,W  ;recupera W
      RETFIE          ;retorna da interrupcao
;==== Fim da Rotina de Interrupcao ====
;====== Inicio da Rotina de Pausa =====
pausap
      movlw  255
      movwf  pausat
pausap1
      movlw  253
      movwf  pausad
pausap2
      nop
      decfsz  pausad,f
      goto   pausap2
      decfsz  pausat,f
      goto   pausap1
      return
;======= Fim da Rotina de Pausa =======
CONTINUA COM ROTINA DE SERIAL.
```

Quadro 6 – Código fonte em Assembly gerado pelo protótipo

Ainda como instrumento de validação, foi disponibilizado uma versão *beta* do protótipo aos acadêmicos da disciplina de Prática em Arquitetura de Computadores onde foram realizados diversos testes para verificar a funcionalidade e detectar erros do software. Alguns dos comentários enviados pelos alunos estão descritos no apêndice A.

Através do estudo de caso apresentado, pôde-se verificar que os objetivos e os requisitos do protótipo foram atingidos com sucesso.

Para demonstrar as diferenças existentes entre o protótipo desenvolvido e o trabalho correlato de Fontanive (1999), foi criada a Tabela 5, onde estão assinaladas com "X" as principais características de cada um.

Tabela 5 – Tabela de comparativo entre os dois trabalhos

Funções	Protótipo desenvolvido	Protótipo Fontanive
Consistência de Variáveis	X	
Impressão em serial	X	
Integração com outros aplicativos	X	
Maior área de edição do fluxograma	X	
Opção de escolha do local para salvar o fluxograma e o código fonte	X	
Opção de imprimir código fonte	X	
Opção para desfazer a seleção de todos objetos selecionados	X	
Opção para excluir os objetos criados	X	X
Opção para selecionar todos os objetos	X	
Opção para trabalhar com variáveis e literais (números)	X	X
Opção para trabalhar com registradores	X	
Opção para trabalhar com variáveis	X	X
Opção para visualizar o conteúdo dos objetos criados sem redimensioná-los	X	
Possibilidade de alterar o código Assembly gerado	X	
Proporcionalidade entre os objetos	X	
Rotina de fim	X	X
Rotina de goto	X	X
Rotina de pausa	X	
Software em língua portuguesa	X	
Tratador de interrupção	X	
Zoom	X	X

Entre as características demonstradas na Tabela 5, pode-se verificar a inexistência de algumas funções importantes no trabalho desenvolvido por Fontanive (1999) que podem inviabilizar o desenvolvimento de um bom programa. Por exemplo, pode ser muito complicado desenvolver programas que controlem interrupções e registradores.

O protótipo aqui desenvolvido contempla algumas das limitações encontradas no trabalho de Fontanive (1999) como as descritas abaixo:

a) gera todos os blocos do fluxograma proporcionais e alinhados e realiza diversas as verificações entre as ligações dos blocos, não permitindo ao usuário criar ligações incorretas, que posteriormente afetarão o código a ser gerado;
b) permite que o usuário escolha o local onde será salvo o fluxograma e o código Assembly;
c) possui a instrução de pausa que é essencial para o controle de algumas ações;

d) possui instrução de interrupção que é essencial para o desenvolvimento de uma boa aplicação;
e) possui opção para imprimir o código gerado;
f) realiza consistência das variáveis que são criadas, não permitindo ao usuário criar variáveis com os mesmos nomes dos registradores utilizados pelo PIC16F84A.

Vale destacar que estas características tornam o protótipo muito mais completo e funcional.

4 CONSIDERAÇÕES FINAIS

Neste capítulo serão descritas as conclusões, algumas vantagens e limitações do protótipo desenvolvido, bem como serão sugeridas extensões para trabalhos futuros.

4.1 CONCLUSÕES

A partir do estudo de caso apresentado, constata-se que a ferramenta introduz uma nova alternativa para o aprendizado da linguagem Assembly e programação de microcontroladores PIC16F84.

Através deste protótipo o usuário poderá resolver alguns problemas que fazem uso de microcontroladores em suas soluções através da construção de fluxogramas, o que torna a programação muito mais prática e amigável. Para operar este protótipo, o usuário deve estar familiarizado com alguns conceitos básicos dos microcontroladores, visto que durante o desenvolvimento do fluxograma poderá ser necessário fazer uso de registradores, rotinas de pausa e setar interrupções para realizar alguns controles no programa.

Os objetivos e os requisitos previamente definidos foram todos atingidos. Portanto, este protótipo já está apto a ser empregado em disciplinas como Prática em Arquitetura de Computadores, Sensores e Atuadores ou Automação e Controle, a fim de facilitar o aprendizado da linguagem Assembly, pois além de gerar todo o código para o usuário, ainda são gerados comentários sobre os comandos em cada bloco e com isto, o usuário pode ir se familiarizando com os comandos do Assembly.

Futuramente, pretende-se realizar algumas melhorias neste protótipo, com o objetivo de torná-lo um produto comercial.

4.2 VANTAGENS E LIMITAÇÕES

Neste item são descritas algumas vantagens e limitações deste protótipo.

Entre as vantagens pode-se citar a possibilidade de se utilizar este protótipo em disciplinas como Prática em Arquitetura de Computadores, Sensores e Atuadores ou Automação e Controle e, apesar de possuir algumas limitações (que serão descritas mais à

frente), o protótipo gera o código corretamente e auxilia no aprendizado da linguagem Assembly, pois os comandos gerados através dos blocos de processo e condição no fluxograma possuem comentários sobre as linhas de código Assembly que são geradas.

Este protótipo apresenta algumas limitações as quais encontram-se descritas nos itens abaixo:

a) ao excluir um bloco de goto, não exclui os *labels* criados nos objetos de origem e destino;
b) não carrega as variáveis cadastradas e as definições de entrada/saída quando abre um fluxograma existente;
c) não permite excluir objetos e conexões a partir da tecla *DELETE*;
d) não permite imprimir o fluxograma, devido a limitações do componente ExpressFlowChart (DEVELOPER, 2004);
e) permite ao usuário criar uma conexão sem destino.

É importante salientar que estas limitações não inviabilizam a utilização do protótipo.

4.3 EXTENSÕES

Como extensões para este trabalho sugere-se:

a) demonstrar a execução passo a passo do fluxograma através de animações;
b) disponibilizar a geração do código para outros modelos de microcontroladores;
c) implementar recursos de macros para dispositivos;
d) imprimir o fluxograma;
e) incluir opção para trabalhar com operações aritméticas nos blocos de condição;
f) integração com o montador;
g) interface com outros periféricos, por exemplo, um *display* de cristal líquido (LCD).

REFERÊNCIAS BIBLIOGRÁFICAS

CARES, Paula Lorena Lovera. **Ambiente para teste de mesa utilizando fluxogramas.** 2002. 92 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Educação Superior de Ciências Tecnológicas, da Terra e do Mar, Universidade do Vale do Itajaí, Itajaí.

DEVELOPER, Express. **Improving then Developer Experience**, Las Vegas, 2004. Disponível em: <http://www.devexpress.com/?section=/products/VCL/ExFlowChart>. Acesso em: 31 mar. 2004.

ELECTRONICS, Labcenter. **Proteus**, Grassington, 2004. Disponível em: <http://www.labcenter.co.uk/>. Acesso em: 25 maio 2004.

EZIMERCHANT, Professional. **Celestial Horizons Pyt Ltd**. Research, Innovation and Development, Melbourne, 2003. Disponível em: < http://www.celestialhorizons.com>. Acesso em: 20 maio 2004.

FONTANIVE, Drayton Roberto. **Protótipo de editor fluxogramático com interface visual para geração de código para o microcontrolador PIC16C84 da Microchip Technology**. 1999. 68 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FURLAN, Jose Davi. **Modelagem de objetos através da UML-The Unified Modeling Language**. São Paulo: Makron Books, 1998. 329 p.

HEMERA, Technologies. **SmartDraw**, San Diego, [2004]. Disponível em: <http://www.smartdraw.com>. Acesso em: 30 mar. 2004.

JACINTO, André Luiz. **Falhas no seu programa** [mensagem pessoal]. Mensagem recebida por <eduardo.pinheiro@senior.com.br> em 14 maio 2004.

MICROCHIP, Trademarks and Patents. **A Leading Provider of Microcontroller & Analog Semiconductors,** Massachusetts, 2003. Disponível em: <http://www.microchip.com/download/tools/picmicro/icds/mplabicd/51184d.pdf>. Acesso em: 20 nov. 2003.

MICROCHIP, Trademarks and Patents. **A Leading Provider of Microcontroller & Analog Semiconductors,** Massachusetts, 2004. Disponível em: <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1475&category=devSoftware>. Acesso em: 21 maio 2004.

MULTIMIDIA, Matrix. **PICmicro products**, Halifax, [2004?]. Disponível em: <http://www.matrixmultimedia.co.uk/picmicro/picprods_1.htm>. Acesso em: 25 mar. 2004.

PIRES, Alexandro Santos. **Bugs do Fluxopic** [mensagem pessoal]. Mensagem recebida por <eduardo.pinheiro@senior.com.br> em 14 maio 2004.

SAVI, Edson. **Fluxopic – Sugestões e erros** [mensagem pessoal]. Mensagem recebida por <eduardo.pinheiro@senior.com.br> em 19 maio 2004.

SILVA JUNIOR, Vidal Pereira da. **Microcontroladores PIC**: teoria e prática. São Paulo: Erica, 1998. 140 p.

SOUZA, David José de. **Desbravando o PIC**. São Paulo: Érica, 2002.

TENENBAUM, Aron M.; LANGSAM, Yedidyah; AUGENSTEIN, Moshe J. **Estrutura de dados usando C**. 1. ed. São Paulo: Makron Books, 1995.

APÊNDICE A – Comentários sobre os testes realizados no protótipo

Este apêndice apresenta alguns comentários e sugestões de acadêmicos da disciplina de Prática em Arquitetura de Computadores que realizaram diversos testes de validação em uma versão *beta* do protótipo.

Abaixo estão os comentários envidados por e-mail:

"Quando é selecionado um objeto, é possível excluí-lo no menu, mas como padrão, a tecla DELETE também deveria acionar o mesmo método de exclusão do objeto selecionado" (PIRES, 2004).

"Ferramenta muito interessante, auxilia no aprendizado dos comandos da linguagem Assembly..." (PIRES, 2004).

"Deveria haver opções no botão direito do mouse, como excluir, mover, modificar..." (JACINTO, 2004).

"Poderia ter opções de voltar (ctrl+z), copiar(ctrl+c) e colar(ctrl+v)" (JACINTO, 2004).

"O protótipo está excelente, é uma ótima ferramenta e pode auxiliar na resolução dos execícios da matéria. (SAVI, 2004).

"Menos de 100% da janela não deixa adicionar processos, condições, etc" (SAVI, 2004).