UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE HARDWARE E SOFTWARE PARA CAPTURA E VISUALIZAÇÃO DE IMAGENS COMPARTILHADAS VIA INTERFACE DIGITAL SERIAL DIFERENCIAL BALANCEADA

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ÂNGELO DIAS DOS SANTOS

BLUMENAU, JULHO/2002

2002/1-10

PROTÓTIPO DE HARDWARE E SOFTWARE PARA CAPTURA E VISUALIZAÇÃO DE IMAGENS COMPARTILHADAS VIA INTERFACE DIGITAL SERIAL DIFERENCIAL BALANCEADA

ÂNGELO DIAS DOS SANTOS

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Antônio Carlos Tavares — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Orientador

_____

Prof.

_____

Prof.

Dedico este trabalho à minha família, em especial À Mara e à Larissa, pelo carinho e compreensão constantes.

AGRADECIMENTOS

O homem é uma ilha, cercado por águas de onde deve tirar o seu sustento. E é por isso que gostaria de agradecer a todos que, seja com uma crítica construtiva, uma palavra de apoio ou atitudes concretas, sustentaram a realização deste trabalho.

Agradecimentos especiais ao professor e orientador Antônio Carlos Tavares, pela cumplicidade de idéias e pelo grande apoio e palavras de motivação; ao professor Miguel Alexandre Wisintainer, pelos esclarecimentos e sugestões; a todos os professores que participaram de minha formação acadêmica; e finalmente, a todos amigos conquistados nessa jornada, pois certamente influenciaram positivamente a execução deste.

A todos da empresa DWA, representada pelo Sr. Paulo Lange e pelo Sr. Luis Osvaldo Urbano, responsáveis diretos por essa obra.

A meus familiares, amigos e colegas, minha gratidão pelo eterno apoio.

"Usa sua obra para mostrar a você mesmo quem você é."

Paulo Coelho, escritor brasileiro

"Se fechares a porta a todos os erros, a verdade ficará do lado de fora."

Rabindranath Tagore, poeta indiano

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

CFTV	-	Circuito Fechado de Televisão
CI	-	Circuito Integrado
CISC	-	*Complex Instruction Set Computer*
CMOS	-	Complementary Metal Oxide Semiconductor
EEPROM	-	*Electrically Erasable Programable Read Only Memory*
EPROM	-	*Erasable Programable Read Only Memory*
fps	-	*frames* por segundo
FSR	-	*File Select Register*
Gbps	-	Gigabits por segundo
I²C	-	*Inter Integrated Circuit*
LCD	-	*Liquid Cristal Display*
Mbps	-	Megabits por segundo
MSSP	-	*Master Synchronous Serial Port*
RAM	-	*Random Access Memory*
RISC	-	*Reduced Instruction Set Computer*
ROM	-	*Read Only Memory*
RS	-	*Recommended Standard*
SFR	-	*Special Function Register*
SPI	-	*Serial Port Interface*
USART	-	*Universal Synchronous Asynchronous Receiver/Transmiter*
USB	-	*Universal Serial Bus*

RESUMO

Este trabalho tem por objetivo principal especificar e implementar um protótipo de hardware e software para compartilhamento de imagens via interface serial diferencial balanceada, visando a implementação de um sistema de monitoramento para automação residencial. O protótipo deverá capturar imagens em diversos pontos usando câmeras digitas, transmitir as imagens via interface serial até pontos de monitoramento, onde serão visualizadas em monitores de cristal líquido

ABSTRACT

The main objective of this work is to specify and to implement a hardware and software prototype that shares images by differential balanced serial interface, intending a home automation systems application. The prototype shall be acquire images in several points using digital cameras, sending them to control points to be visualized in liquid crystal displays.

1 INTRODUÇÃO

Depois de sua consolidação na indústria e, mais recentemente, no comércio, a automação enfim chega ao ambiente doméstico. Tendências e tecnologias já são discutidas em feiras e congressos internacionais. Eventos estes inclusive já sendo realizados no Brasil (As Casas, 2001), mostrando que a automação predial residencial - ou Domótica - já é realidade e campo promissor para pesquisa e desenvolvimento a nível nacional.

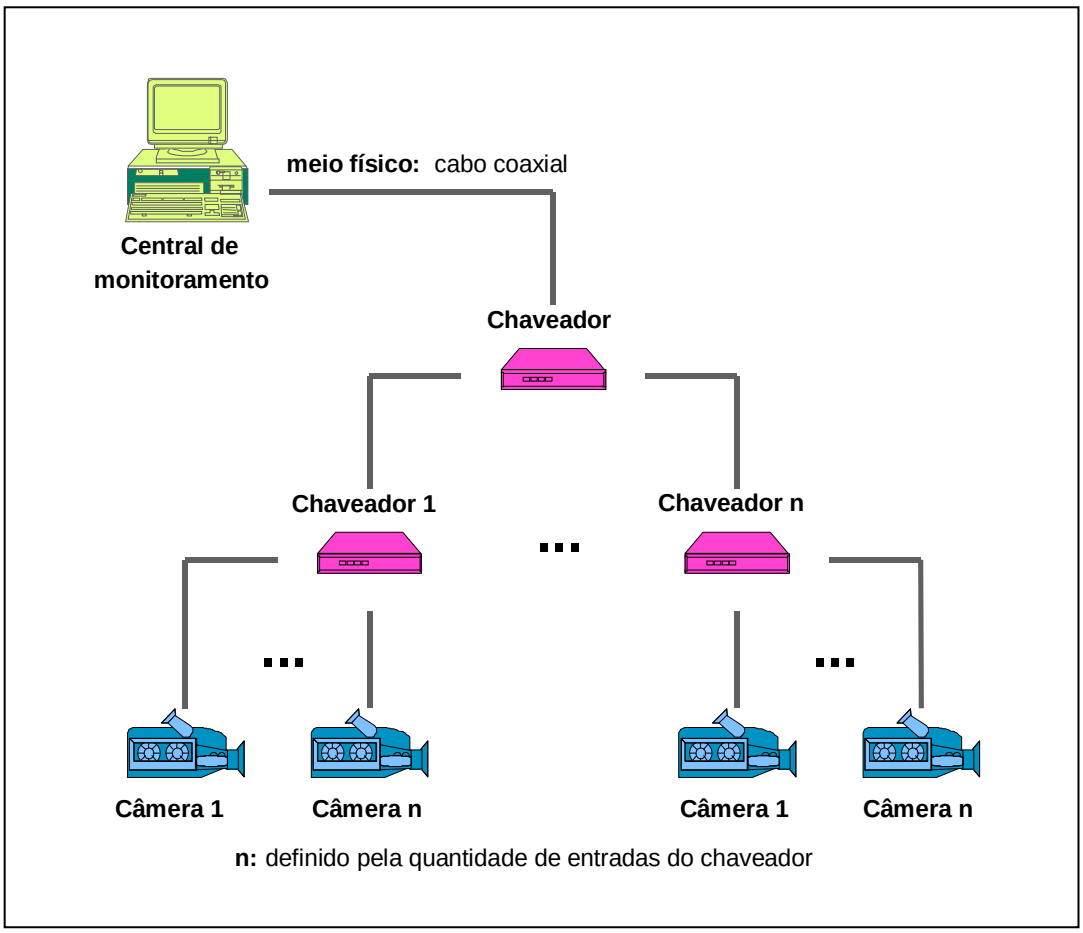
Numa avaliação feita por Marte (1995), pode-se verificar a várias tecnologias que já estão sendo aplicadas para a automação e gerenciamento de residências. Áreas como consumo de água potável, consumo de energia elétrica, climatização, comunicação, entretenimento e segurança são alvos de uma ampla gama de dispositivos e um sem número de soluções visando melhorar a qualidade de vida.

As soluções apresentadas possuem maior ou menor grau de dificuldade para implementação, mas alguns problemas são bem evidentes. Em Aureside (2001) salienta-se que tanto prédios existentes quanto projetos de novas edificações não contemplam o conceito de automação, principalmente no que diz respeito à infra-estrutura elétrica (cabeamento e definição de equipamentos). Além disso, a maior parte dos equipamentos de automação doméstica não ficam obrigatoriamente incorporados ao imóvel, normalmente sendo levados pelo proprietário quando ocorre mudança de endereço. O custo, nesse caso, acaba sempre falando mais alto.

Tome-se como exemplo os circuitos fechados de televisão (CFTV), muito usados como ferramentas auxiliares na automação de segurança predial (fig. 1). Fica flagrante o uso do microcomputador como central de monitoramento e, implicitamente, o uso de placas de captura de vídeo para a conexão com essa central.

Apesar de Marte (1995) ter sido citado como fonte de referência, Aureside (2001) também descreve um sistema de monitoramento baseado nessa tecnologia. A tendência, porém, é o uso de equipamentos para a digitalização de imagens, o que reduziria o custo com cabeamento e dispositivos intermediários. Mas esses equipamentos ainda são caros e ainda atrelados a computadores pessoais como central de monitoramento, formando sistemas de excelente performance mas também bastante complexos para um usuário doméstico.

FIGURA 1 – SISTEMA ANALÓGICO DE CFTV



Adaptado de Marte (1995)

É neste cenário que se enquadra a implementação de um protótipo de sistema de vigilância e monitoramento de custo reduzido, fácil implementação e manutenção. A proposta é de um sistema composto por câmeras digitais, para captura de imagens, e pequenos módulos contendo telas de cristal líquido (LCD ou *liquid cristal display*), para visualização. A interligação destes dispositivos é feita por apenas um par de condutores, numa configuração multiponto, através de interface serial com tecnologia diferencial balanceada.

Tanto hardware (parte física do sistema) quanto software (maneira como o hardware irá se comportar) são dedicados. Isto é, desenvolvidos especificamente para a função a que se destinam. As câmeras digitais, facilmente encontradas em módulos simples e acessíveis, podem ser instaladas em diversos pontos de controle, dentro e fora da residência. Da mesma forma, os módulos LCD, ou simplesmente *displays*, são encontrados em diversas

configurações. A especificação da rede de comunicação serial multiponto (vários equipamentos ligados ao mesmo meio físico) e da tecnologia diferencial balanceada (explicada no capítulo 2) , como padrão para a transmissão de sinais no meio físico, deve-se tanto ao reduzido custo de implementação como à excelente performance a nível de velocidade e imunidade a ruído – fatores essenciais para a transmissão de imagens.

1.1 OBJETIVOS DO TRABALHO

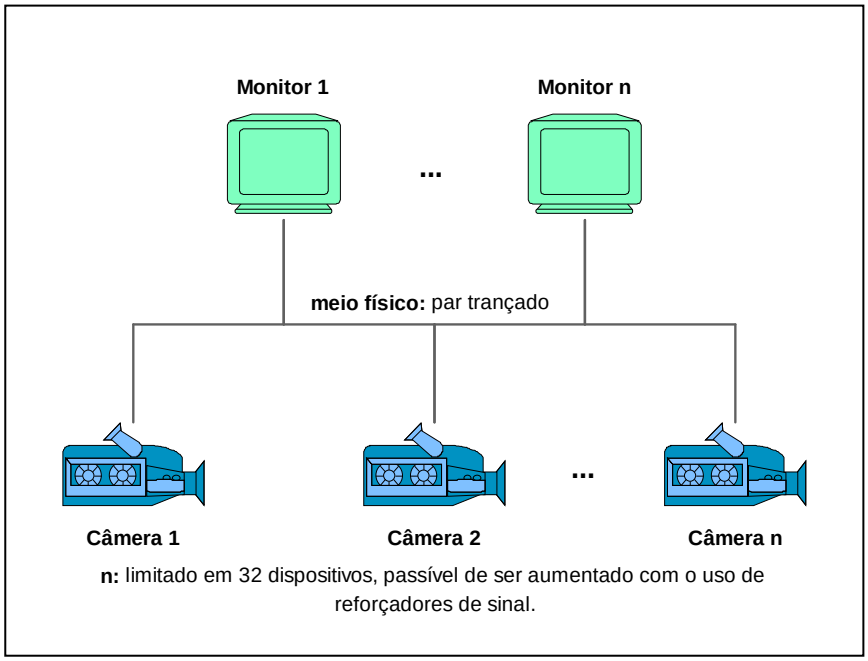
O objetivo do trabalho é construir um protótipo de hardware e software para compartilhamento de imagens via interface serial diferencial balanceada visando a implementação de um sistema de vigilância/monitoramento para automação residencial.

Os objetivos específicos do trabalho são:

a) capturar imagens usando câmeras digitais;
b) tratar as imagens para transmissão via interface serial diferencial balanceada;
c) visualizar as imagens remotamente em monitores de cristal líquido.

Como objetivo secundário, pode-se apontar o aperfeiçoamento da programação em linguagem de montagem (*assembly*). Este aspecto é de grande relevância para o trabalho, que divide-se em três partes básicas: monitores, câmeras e rede de comunicação de dados (fig. 2).

FIGURA 2 – O PROTÓTIPO: CÂMERAS, REDE E MONITORES



1.2 ESTRUTURA DO TRABALHO

O texto está organizado de forma a possibilitar uma visão geral das teorias e tecnologias utilizadas, buscando aprofundamento apenas em questões que fogem do censo comum na área de ciências da computação. Alguns destaques serão dados a assuntos que, aparentemente corriqueiros, tenham representado considerável ganho de conhecimento durante o trabalho de pesquisa ou mesmo implementação.

No capítulo 2 encontra-se a revisão bibliográfica, disposta por assunto, das diversas tecnologias envolvidas. É feita uma introdução ao assunto, para em seguida haver um detalhamento maior a respeito das técnicas investigadas em tópicos relevantes a nível de implementação.

Devido à sua importância, as ferramentas utilizadas no desenvolvimento de hardware e software, ganharam um capítulo à parte. O capítulo 3 faz uma explanação das funções e modo de funcionamento dessas ferramentas, a fim de permitir uma melhor avaliação pela separação dos conteúdos.

Especificação, implementação e testes são descritos no capítulo 4. A disposição das informações visa salientar a maneira como foi desenvolvido o trabalho. Dispostas em ordem cronológica de implementação estão rede de comunicação, visualização e captura da imagem, que foram assim incorporados ao protótipo.

O capítulo 4 destaca, então, as conclusões alcançadas. Traz também sugestões para o desenvolvimento de trabalhos correlatos, a quem possa eventualmente interessar-se pelo mundo do hardware.

2 FUNDAMENTAÇÃO TEÓRICA

Este trabalho envolve tecnologias que visam a integração entre hardware e software, o que requer certo domínio em eletrônica básica. A escolha, a interligação de componentes e a compatibilização dos sinais elétricos necessários ao funcionamento do circuito devem ser bem entendidos para que o conjunto funcione adequadamente. Capacidade de manipular e entender instrumentos de medição elétrica também é relevante. Na verdade, é um detalhe pertinente a todo programador: conhecer o ambiente onde seu software irá ser executado, não importando o nível da linguagem utilizada.

Pressupondo-se então fundamentação básica em eletrônica, passa-se a tratar do que cabe à área de ciências da computação. Arquitetura de microcontroladores (incluindo linguagem de máquina), comunicação de dados, captura e visualização de imagens são técnicas utilizadas na implementação e que caracterizam aspectos relevantes deste trabalho.

Antes, porém, de proceder com o estudo das técnicas supracitadas, é necessário que se destaque o trabalho desenvolvido por Klitzke (1999) que também propõe a captura de imagens e sua transmissão via interface diferencial balanceada, buscando visualização a partir de um computador pessoal. A diferença básica é a utilização de hardware dedicado na tarefa de visualizar a imagem, através de módulos LCD.

Além disso, busca-se implementar o compartilhamento com múltiplos pontos de captura e visualização, usando para tanto dispositivos com maior capacidade de processamento – com características e linguagem de programação diferenciadas das apresentadas em Klitzke (1999).

2.1 ARQUITETURA DE MICROCONTROLADORES

Os microcontroladores são os responsáveis pela tarefa de gerenciar a aquisição, a visualização e o compartilhamento das imagens. A correta integração entre rede, módulos de aquisição e visualização de imagem somente será possível se sua programação for eficiente, o que representa o real mérito do trabalho.

Para que não haja dúvidas quanto ao uso dos termos *microprocessadores* e *microcontroladores*, são relatados alguns pareceres sobre o assunto. Silva Júnior (1990), por

exemplo, aponta o microprocessador como sendo um componente eletrônico que, devido às suas modernas técnicas de fabricação, consegue efetuar com rapidez várias funções e operações lógicas e aritméticas, sob controle de um programa externo que dita para a máquina a seqüência das funções e os operandos a serem utilizados.

Muitos autores usam o termo *microcontroladores* para componentes que possuem agregadas funções extras à unidade de processamento. Estes dispositivos, também chamados de "microcomputadores de um só *chip*" (Silva Júnior, 1997), são dotados de inteligência programável e utilizados no controle de processos lógicos. Segundo Souza (2000), diferem dos microprocessadores por apresentarem periféricos dedicados a aplicações específicas como contadores de eventos, temporizadores, conversores analógico/digital, portas de entrada/saída, memórias RAM e ROM, canais de comunicação paralela e de comunicação serial.

2.1.1 MICROCONTROLADORES

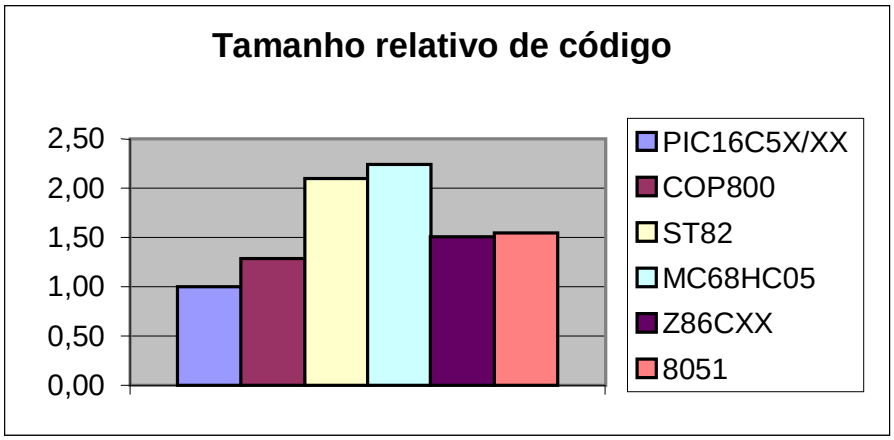
Existe uma infinidade de opções de microcontroladores, de vários fabricantes, disponível atualmente no mercado. Cada um possuindo diferentes características de arquitetura, velocidade de processamento, conjunto de instruções, capacidade de memória e periféricos para os mais diversos fins.

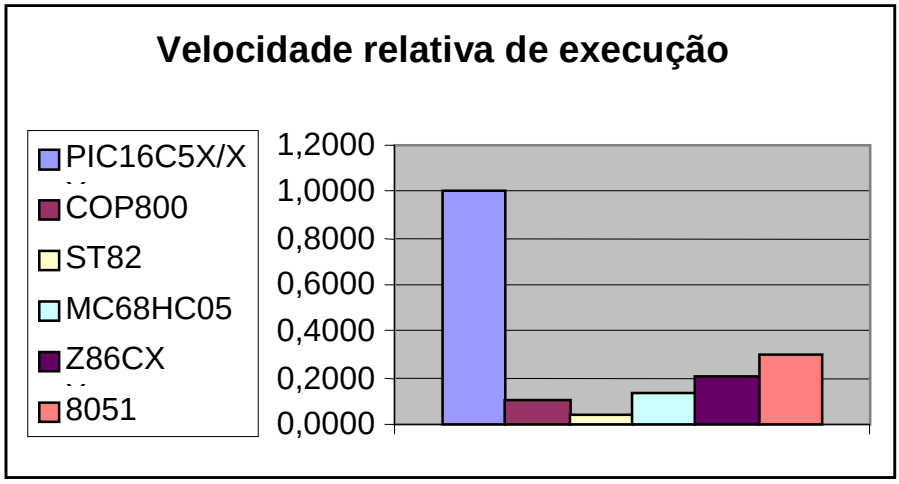
Os fabricantes oferecem atrativos como ambientes de desenvolvimento, que incluem simuladores e emuladores, além de gravação/emulação *in-circuit*, para que os projetistas de hardware e programadores especifiquem e usem seus produtos. Esses aplicativos, muitas vezes gratuitos, são ferramentas que facilitam muito a tarefa de programação.

Cabe ao projetista de hardware determinar as variáveis que serão analisadas na hora de optar por um fabricante, família ou modelo de microcontrolador. Em Microchip (1996), uma comparação é feita entre diversos modelos, graficamente representada na fig. 3.

FIGURA 3 – COMPARAÇÃO ENTRE FAMÍLIAS DE MICROCONTROLADORES

Modelo	Fabricante	Clock (MHz)
PIC16C5X/XX	Microchip	20
COP800	National	20
ST82	SGS-Thomson	8
MC68HC05	Motorola	4,2
Z86CXX	Zilog	12
8051	Intel	20





Adaptado de Microchip (1996)

Para ilustrar essa diversidade de modelos, o presente capítulo discorre sobre uma família de microcontroladores que tem se tornado muito popular nos últimos anos. Os microcontroladores da linha PIC, da Microchip. Estes diferem bastante dos já consagrados

microcontroladores da família MCS-51 da Intel, bastante utilizados em trabalhos acadêmicos – Klitzke (1999) é um exemplo. Documentação a respeito do 8051, o termo genericamente utilizado para referenciar microcontroladores da família MCS-51, pode ser encontrada em Intel (1981) e Silva Júnior (1990).

2.1.2 FAMÍLIA DE MICROCONTROLADORES PIC

Os microcontroladores da linha PIC, segundo Souza (2000), diferem da linha 8051 principalmente na sua estruturação interna, já que utilizam-se de barramentos separados para dados e instruções. Esta arquitetura, denominada *Harvard*, determina que memória e barramento de dados (8 bits) sejam separados da memória e barramento de instruções (12 a 16 bits - dependendo do modelo do microcontrolador).

O fato de os barramentos serem separados permite ao dispositivo trabalhar no modo *pipeline* ou "linha de montagem", isto é, o microcontrolador executa uma instrução enquanto busca a próxima a ser executada. A conseqüência dessa estruturação interna é que a maioria das instruções são executadas em apenas um ciclo de máquina, excetuando-se as que provocam desvios na execução. Estas ocupam dois ciclos da máquina, sendo que um é usado para a limpeza do *pipeline* (Souza, 2000).

Ainda segundo Souza (2000), para poder trabalhar em *pipeline*, a instrução já deve conter o *opcode* (código da instrução), o dado e o local onde ela vai operar (quando necessário) numa única palavra de memória. Aliado ao fato de que o barramento tem no máximo 16 bits, isto faz com que as instruções devam ser mais simples, reduzindo-as a uma quantidade aproximada de 35 instruções (conforme o modelo do PIC utilizado).

Reduced Instruction Set Computer (RISC) – computador com conjunto reduzido de instruções, esta é a nomenclatura utilizada para referenciar os microprocessadores/microcontroladores que adotam esta filosofia. Em contrapartida, Complex Instruction Set Computer (CISC) são dispositivos que, de modo análogo ao 8051, possuem um amplo conjunto de instruções. No quadro 1 podemos ver um comparativo entre CISC e RISC.

Em Silva Júnior (1997) encontra-se destacada ainda outra característica importante: nos dispositivos PIC, os ciclos de máquina são divididos em quatro fases. Operando com um

clock de apenas 4MHz cada ciclo de máquina é executado em 1 μs (0,000001 segundo). Logo, a maior parte das instruções, conforme já citado, levaria este mesmo período para ser executada.

QUADRO 1 – COMPARAÇÃO ENTRE COMANDO CISC E RISC

Exemplo de comando	Somar 12 ao registro 24	
Execução CISC	Somar (operação) 12 (operando) 24 (endereço)	3 bytes
Execução RISC	Somar (operação) + 12 (operando) + 24 (endereço)	1 palavra de 14 bits

Adaptado de Silva Júnior (1997)

Microchip (2001) disponibiliza manual completo dos modelos PIC16F87X, que são microcontroladores de 8 bits com memória FLASH. Dentre eles, o PIC16F877 chama a atenção por possuir memória de programa c/ 8K palavras de 14 bits, além de cinco portas de I/O e duas portas de comunicação serial.

2.1.3 MICROCONTROLADOR PIC16F877

Este é um microcontrolador que possui praticamente todas as opções disponíveis em termos de periféricos. A descrição dos pinos do componente pode ser vista no anexo I e as partes principais são detalhadas no decorrer deste capítulo. Na fig. 4 encontra-se o diagrama de blocos do microcontrolador, visto em Microchip (2001), que faz um resumo do mesmo:

a) CPU de alta performance RISC (35 instruções), com clock máximo de 20MHz;
b) memória FLASH de programa com 8K palavras de 14 bits;
c) memória de dados: RAM (368 bytes) e EEPROM (256 bytes);
d) 14 pinos de interrupção configuráveis;
e) pilha com 8 níveis;
f) modos de endereçamento direto, indireto e relativo;
g) circuitos e temporizadores de inicialização do sistema para garantir estabilização de

clock e integridade de dados em caso de inicialização do sistema;

h) temporizador *watchdog* com oscilador próprio;

i) proteção de código programável;

j) opções de freqüência de oscilador selecionáveis;

k) programação serial e depuração de erros *in-circuit* usando dois pinos;

l) faixa de alimentação de 2 a 5,5Vcc;

m) capacidade de corrente de 25mA;

n) baixo consumo – em modo *SLEEP* ainda mais econômico;

o) dois temporizadores/contadores de 8 bits e um de 16 bits, com *prescaler*;

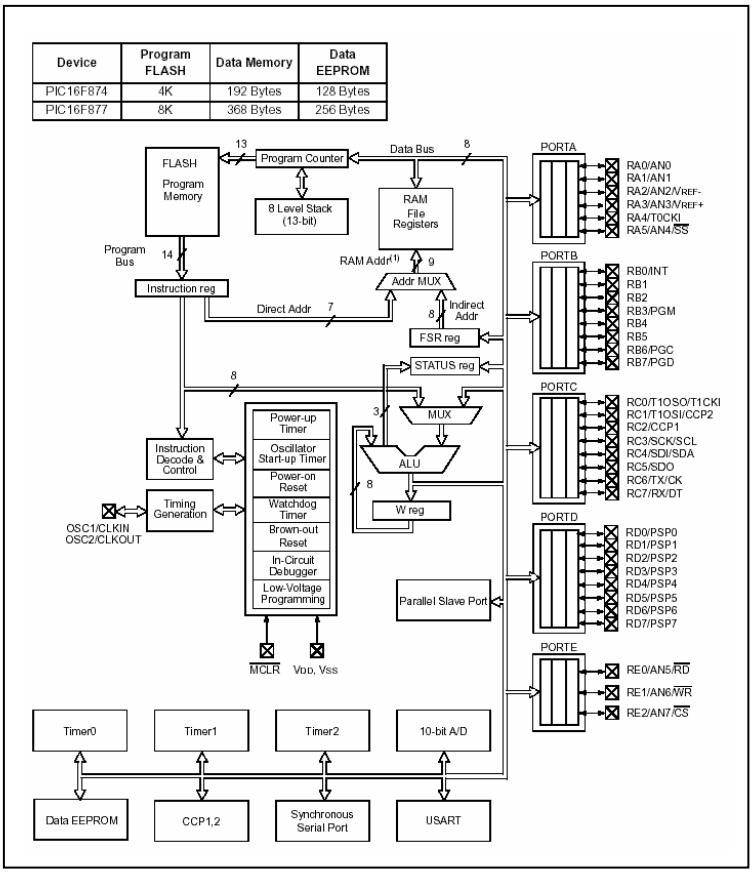
p) dois módulos PWM com resolução de 10 bits;

q) conversor analógico/digital de 10 bits;

r) porta serial síncrona com SPI e I²C e USART com detecção de endereço;

s) porta paralela de 8 bits.

FIGURA 4 – DIAGRAMA DE BLOCOS DO PIC16F877



Adaptado de Microchip (2001)

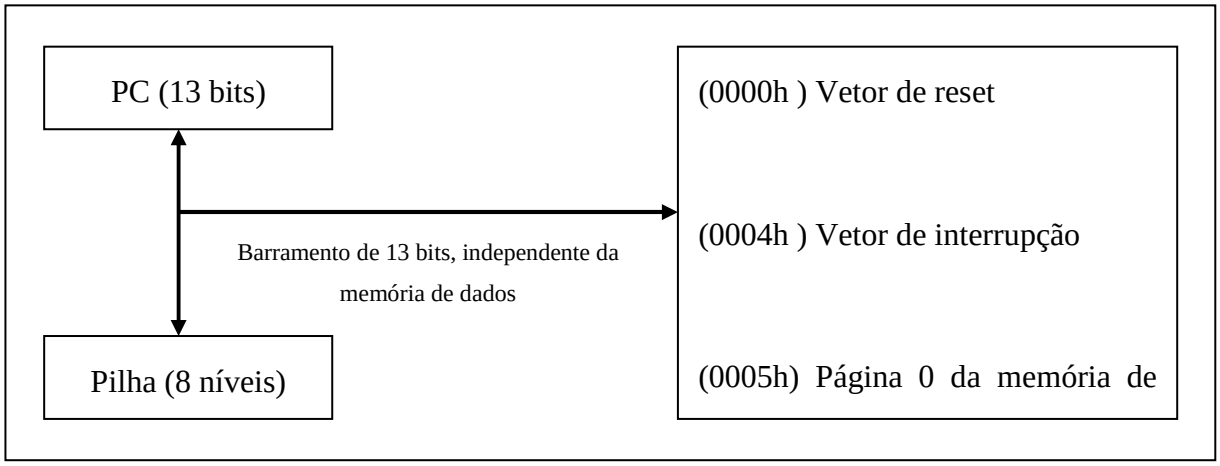
2.1.3.1 ORGANIZAÇÃO DA MEMÓRIA

Existem três blocos de memória no PIC16F877. A memória de programa e a memória de dados, conforme já visto, possuem barramentos separados que podem ser acessados simultaneamente. Existe ainda uma terceira memória, a EEPROM de dados, que consegue manter os dados mesmo sem alimentação (Souza, 2000).

2.1.3.1.1 MEMÓRIA DE PROGRAMA

O microcontrolador possui contador de programa (PC) de 13 bits, capaz de endereçar 8k posições de memória com palavras de 14 bits em memória FLASH. O endereço do vetor de reste é 0000h e do vetor de tratamento para qualquer interrupção é 0004h, conforme visto na fig. 5.

FIGURA 5 – BARRAMENTO DE INSTRUÇÕES DO PIC16F877



Adaptado de Microchip (2001)

A pilha (*stack*) é totalmente separada da memória de programação e não é acessível a nível de usuário, isto é, operações de escrita e leitura não são permitidas. Nos modelos 16F877, a pilha possui oito níveis ou oito posições para armazenamento de endereços de retorno. Isto requer um cuidado extra por parte do programador, pois se mais de oito instruções de desvio ou interrupções simultâneas ocorrerem, o primeiro endereço de retorno será perdido sobrescrito pelo último (Silva Júnior, 1997) (Microchip, 2001).

As interrupções, como já visto, apontam sempre para a mesma posição de memória. Quando ocorrem, devem ser testadas através de *flags* em registradores específicos (vistos

mais adiante) para determinar quem provocou a interrupção e que rotina será chamada para tratamento. Tudo isto deve ser previsto pelo programa.

2.1.3.1.2 MEMÓRIA DE DADOS

A memória de dados está dividida em quatro bancos os quais contém registradores de uso geral e registradores de função especial (*Special Function Registers* – ou SFR´s). Os bits RP0 e RP1 do SFR STATUS são usados para seleção dos bancos de memória, que possuem 128 bytes cada um.

Todos os bancos possuem SFR's, sendo que as locações mais baixas de cada banco são reservadas a estes. Alguns destes registradores especiais são mapeados em mais de um banco, como o STATUS por exemplo, facilitando o acesso e a programação. Logo após os SFR's são implementados os registradores de uso geral, usados como RAM estática.

Os registradores de uso geral podem ser acessados de forma direta ou indiretamente através do registrador especial FSR (*File Select Register*). Os SFR's são registradores usados pela CPU e pelos módulos periféricos para controlar as operações executadas pelo dispositivo. O mapa da memória de dados, contendo registradores de uso geral e de função especial está representado no anexo II.

2.1.3.2 PORTAS DE ENTRADA E SAÍDA

No PIC16F877 são cinco portas de entrada e/ou saída que possuem funções alternativas como pode ser constatado em Microchip (2001). No anexo I podem ser vistas as portas e suas segundas e, muitas vezes, até terceiras funções.

As portas são configuradas através dos registradores TRIS e lidas/escritas através dos registradores PORT respectivos. Assim, TRISB configura a porta B, que é lida/escrita através do registrador PORTB. Para memorizar a regra de configuração das portas, Souza (2000) ensina uma maneira prática: quando é colocado "1" em um bit do TRIS, o pino relacionado a ele é configurado como entrada, e quando recebe "0" é configurado como saída – sendo assim, é fácil associar o "1" a "*In*" e o "0" a "*Out*".

2.1.3.3 USART

A USART (*Universal Synchronous Asynchronous Receiver Transmitter*) ou transmissor/receptor síncrono/assíncrono universal é um dos dois módulos de entrada e saída serial. O outro é o MSSP (*Master Synchronous Serial Port*) muito usado para comunicação com outros periféricos ou microcontroladores no mesmo circuito.

Conhecida também como interface de comunicação serial (SCI), a USART possui capacidade para detecção de endereço (9 bits) e pode ser configurada como:

a) Modo assíncrono (full-duplex);
b) Modo síncrono – mestre (half-duplex);
c) Modo síncrono – escravo (half-duplex).

O bit SPEN (registrador RCSTA, bit <7>) e os bits TRISC<7:6> devem ser setados para configurar os pinos RC6/TX/CK e RC7/RX/DT como TX e RX, respectivamente. Outros SFR's envolvidos em operações da USART são:

a) TXSTA – registro de controle e *status* de transmissão;
b) RCSTA – registro de controle e *status* de recepção;
c) SPBRG – gerador de taxa de transmissão.

O SPBRG é o registrador interno que controla o temporizador responsável por gerar a taxa de transmissão. Usando um cristal de 20MHz, o 16F877 é capaz de alcançar as taxas de 1,25Mbps (um milhão, duzentos e cinqüenta mil bits por segundo) em modo assíncrono e 5Mbps em modo síncrono.

2.1.3.4 CONSIDERAÇÕES SOBRE PROGRAMAÇÃO

O conjunto de 35 instruções do PIC16F877, o mesmo usado pelos outros modelos da linha, encontra-se no anexo III. No que diz respeito a ferramentas de desenvolvimento, o fabricante disponibiliza gratuitamente o software MPLab em sua página da Internet (Microchip, 2000). Este, em conjunto com a linguagem MPAsm (criada para os dispositivos PIC), forma um excelente ambiente de desenvolvimento segundo Souza (2000), que também afirma ser esse um dos principais motivos da popularização desses microcontroladores.

O MPLab junta no mesmo ambiente o gerenciamento de projetos, a compilação, a simulação, a emulação e a gravação do *chip*. Os dois últimos, porém, somente se usados em conjunto com dispositivos de emulação/gravação compatíveis. O aplicativo é melhor explicado no capítulo 3, apesar de existirem outras ferramentas para desenvolvimento baseadas em linguagens de alto nível, como a linguagem C.

Para complementar o manual do componente, que está em inglês, boa documentação em português é encontrada em Souza (2000) e Silva Júnior (1997). Apesar de estarem baseados em outro microcontrolador, o PIC16F84, as características de ambos são basicamente as mesmas, trazendo estes importante contribuição principalmente com relação a técnicas de programação.

2.2 COMUNICAÇÃO DE DADOS

Para que a imagem coletada pelas câmeras possa chegar ao monitor, uma rede de comunicação de dados deve ser implementada. Os conceitos fundamentais referentes a comunicação de dados são freqüentemente mencionados em diversos trabalhos e são sucintamente expostos em Tafner (1996) e Nunes (1989).

Quanto ao método de transferência de dados, é notório que a transferência paralela leva vantagem sobre a serial quando as distâncias não ultrapassem poucos metros. Porém, no caso da proposta, as distâncias podem envolver dezenas de metros, o que viabiliza a transferência serial em detrimento à paralela.

Dos padrões seriais analisados então, o *Universal Serial Bus* (USB) e o *FireWire* são as tecnologias emergentes. Estas especificações prometem taxas de 1,5Mbps até 400Mbps, atualmente, com previsão de alcançar a faixa dos Gbps (bilhões de bits por segundo) em poucos anos. Entretanto, Torres (2000) deixa transparecer que estas especificações esbarrariam em problemas como falta de documentação e curto alcance, já que foram criados para a conexão de periféricos aos computadores.

Entretanto, existem especificações de padrões seriais criados há alguns anos mas que continuam sendo largamente utilizados. É o caso dos padrões *Recommended Standard* (RS), que são normas criadas para a especificação de interfaces de comunicação de dados. Com base nas informações da tabela 1, pode-se verificar que o padrão RS-485 seria o mais

recomendado para a criação de uma rede multiponto (até 32 pontos) que necessite de distâncias de até 4000 pés (cerca de 1200m) e que necessite de boa taxa de transferência de dados. Comparação entre outros padrões de interface pode ser verificada em Klitzke (1999), onde fica latente a opção da RS-485 para longas distâncias.

TABELA 1 – ESPECIFICAÇÕES DE PADRÕES RS

SPECIFICATIONS		RS232	RS423	RS422	RS485
Mode of Operation		SINGLE -ENDED	SINGLE –ENDED	DIFFER-ENTIAL	DIFFER-ENTIAL
Total Number of Drivers and Receivers on One Line		1 DRIVER 1 RECVR	1 DRIVER 10 RECVR	1 DRIVER 10 RECVR	1 DRIVER 32 RECVR
Maximum Cable Length		50 FT.	4000 FT.	4000 FT.	4000 FT.
Maximum Data Rate		20kb/s	100kb/s	10Mb/s	10Mb/s
Maximum Driver Output Voltage		+/-25V	+/-6V	-0.25V to +6V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	**Loaded**	+/-5V to +/-15V	+/-3.6V	+/-2.0V	+/-1.5V
Driver Output Signal Level (Unloaded Max)	**Unloaded**	+/-25V	+/-6V	+/-6V	+/-6V
Driver Load Impedance (Ohms)		3k to 7k	>=450	100	54
Max. Driver Current in High Z State	**Power On**	N/A	N/A	N/A	+/-100uA
Max. Driver Current in High Z State	**Power Off**	+/-6mA @ +/-2v	+/-100Ua	+/-100uA	+/-100uA
Slew Rate (Max.)		30V/uS	Adjustable	N/A	N/A
Receiver Input Voltage Range		+/-15V	+/-12V	-10V to +10V	-7V to +12V
Receiver Input Sensitivity		+/-3V	+/-200mV	+/-200mV	+/-200mV
Receiver Input Resistance (Ohms)		3k to 7k	4k min.	4k min.	>=12k

Fonte: Smith (1999).

2.2.1 INTERFACE RS-485

A RS-485, como padrão de comunicação, especifica as características elétricas de uma interface serial digital entre circuitos, baseada em tensão diferencial balanceada. O padrão especifica o modo de transmissão assíncrono, mas não determina, por exemplo, o tipo de condutor (normalmente par trançado) nem o protocolo (geralmente proprietário) (Franco, 2001).

Franco (2001) ainda explica que uma interface balanceada é mais vantajosa para altas taxas de transmissão (até 10 Mbps), longas distâncias e ambientes ruidosos. A lógica para este padrão é definida quando um terminal torna-se mais negativo ou mais positivo que o outro. Portando, a tensão diferencial entre os dois terminais permitirá o reconhecimento do bit que está sendo transmitido, se é "0" ou se "1". Convencionalmente, a lógica "1" é reconhecida quando o terminal A do transmissor torna-se mais negativo em relação ao terminal B. A

lógica "0" é identificada quando o terminal A torna-se positivo em relação ao terminal B; é por esse motivo que tem-se maior imunidade a ruídos eletromagnéticos externos.

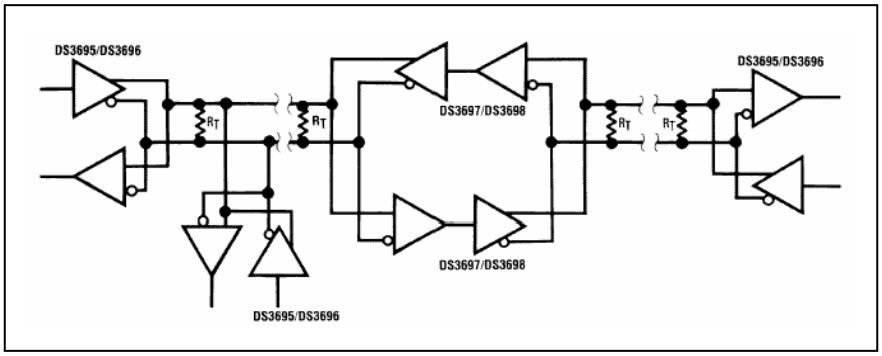
Este padrão determina oficialmente a interligação de até 32 dispositivos (nós de rede), compartilhando um par trançado para transmissão e recepção (configuração half-duplex multiponto). Todavia, já existem no mercado dispositivos (circuitos integrados) reforçadores de sinal, capazes de aumentar a capacidade da rede para a conexão de 128 a 255 nós. Além disso, a taxa de 10Mbps é alcançada a no máximo 15m, enquanto que a 1200m (aproximadamente 4000 pés) a taxa cai a 100Kbps (Integrity Instruments, 2001).

Existem, porém, dois cuidados a serem tomados com relação à instalação da rede em si. São cuidados que previnem problemas elétricos:

a) para garantir a não existência de ondas estacionárias que podem causar falha de comunicação, um resistor de terminação, ou terminador, de 120 ohms deve ser colocado nas extremidades da rede, conforme fig. 6;

b) para maior segurança pessoal e do equipamento, o cabo deve possuir malha que deverá ser aterrada em apenas uma das extremidades (Atos, 2000).

FIGURA 6 – LIGAÇÃO DO TERMINADOR NA REDE 485



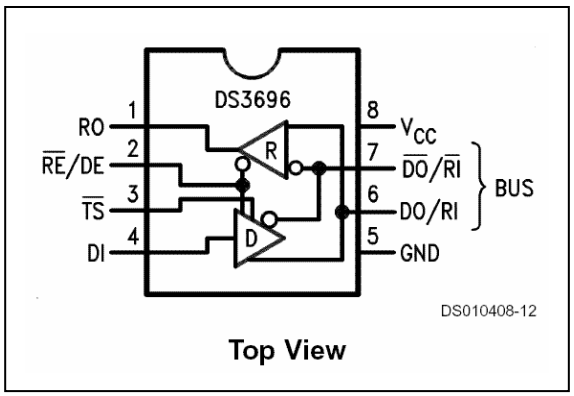
Fonte: National Semiconductor (1998).

Por ser simples e, sobretudo, por apresentar boa imunidade a interferências eletromagnéticas, foi amplamente adotada para a construção de redes multiponto em ambientes industriais, conforme Cunha (2000). Para suprir esse mercado, surgiram um grande número de circuitos integrados (CI´s). O MAX485 (da Maxim), o DS75176B e o DS3696 (ambos da National Semiconductor) são exemplos de dispositivos desenvolvidos para este fim.

2.2.2 CIRCUITO INTEGRADO INTERFACE RS-485

O DS3696 é um dispositivo desenvolvido em concordância com a norma EIA RS485 para transmissão multiponto de dados (National Semiconductor, 1998). É um transmissor diferencial de alta velocidade cuja descrição dos pinos pode ser vista na fig. 7. Outros membros da família são os já citados reforçadores de sinal, ou repetidores, DS3697 e DS3698.

FIGURA 7 – DESCRIÇÃO DOS PINOS DO DS3696



Fonte: National Semiconductor (1998).

O circuito é de fácil compreensão. A alimentação do *chip* se dá através dos pinos 5 e 8. Os pinos RO e DI representam a interface lógica e são diretamente conectadas aos pinos RX e TX de microcontroladores com módulo de comunicação serial (o 8051 e alguns integrantes da família PIC, por exemplo). O controle de transmissão/recepção (half-duplex) é feito através do pino DE/RE, onde um sinal de nível alto (1) habilita o dispositivo a transmitir, enquanto o nível baixo (0) inibe a transmissão e põe o DS3696 em estado de recepção.

Os pinos 6 e 7 formam a interface diferencial do circuito e são conectados entre si. Ou seja, todos os pinos 6 de todos os DS3696 presentes na rede são interligados. Da mesma forma, todos os pinos 7 são também interligados.

Em tempo, quando um DS3696 transmite, o sinal que recebe em sua interface lógica é convertido para tensão diferencial aplicada nos pinos 6 e 7, que serão recebidos pelos dispositivos do outro lado e reconvertidos em sinal digital.

O cuidado que deve ser tomado pelo desenvolvedor é quanto ao controle feito no DE/RE e quanto ao dado lido/escrito nos pinos RX/TX, pois tudo o que acontece entre esses circuitos de interface pode ser desconsiderado a nível de implementação de software.

2.3 CAPTURA DE IMAGENS

Com o advento das câmeras digitais, o processo de aquisição da imagem pôde ser transformado em mais uma fonte de dados diretamente manipuláveis por um sistema computacional, como em aplicações na área médica, por exemplo (Informática Médica, 1998).

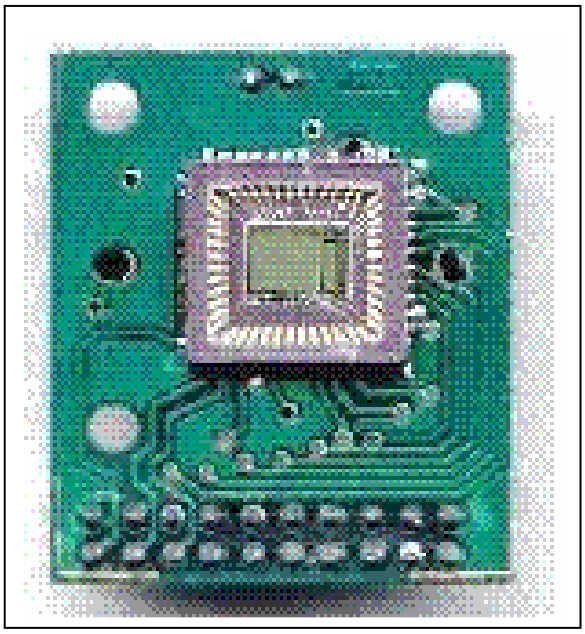
Apesar de ainda longe da excelência da imagem analógica, já existem câmeras digitais com resolução de 1800 por 1200 *pixels* – longe dos iniciais 640 por 800 *pixels* dos primeiros modelos de câmeras fotográficas comerciais de 1997. A qualidade de imagem depende de vários fatores, mas o principal deles é a resolução do sensor, dispositivo sensível à luz formado por diversas células capazes de armazenar informação. De acordo com (Nagano, 2000), quanto maior o número de elementos, maior a resolução e, conseqüentemente, melhor detalhamento da imagem captada. Klitzke (1999) fornece maiores detalhes sobre o funcionamento destes dispositivos.

2.3.1 MÓDULO DE CÂMERA DIGITAL M4088

Abstraindo o modo como as câmeras digitais capturam as imagens, Comedia (1999) apresenta detalhadamente o módulo M4088, que é uma câmera monocromática de ¼” com saída digital que usa o sensor de imagem OV5017 (Omnivision Technologies, 1998).

O módulo é mostrado na fig. 8, sem a lente, para dar ênfase ao sensor, que utiliza tecnologia CMOS. Esta tecnologia em conjunto com uma interface digital de fácil uso, faz deste conjunto uma solução de baixo custo para aplicações de vídeo de boa qualidade.

FIGURA 8 – MÓDULO M4088



Fonte: Comedia (1999)

O M4088 possui uma porta de vídeo digital que fornece fluxo contínuo de dados de 8 bits. As configurações da câmera incluem taxa variável de *frames* (quadros), ajuste de exposição e tamanho da imagem. Todas as configurações são baseadas em registradores. A leitura da imagem e a configuração da câmera é tão fácil quanto o acesso a uma memória externa. A tabela 2 demonstra os sinais necessários para a operação do módulo em modo digital.

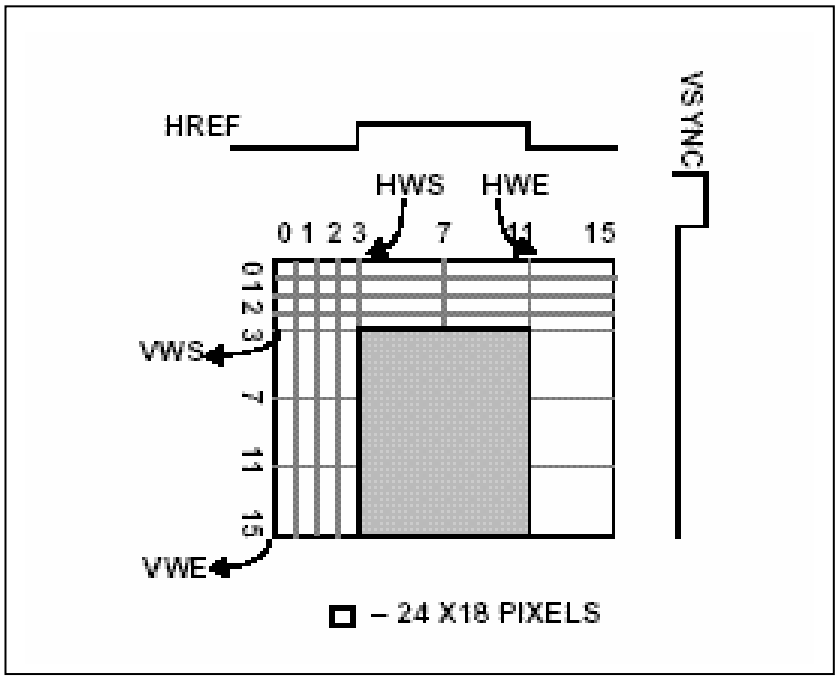
TABELA 2 – DESCRIÇÃO DOS PINOS DO MÓDULO M4088

Pino	**Mnemônico**	**Tipo**	**Função**
1 a 8	D0 a D7	Entrada/saída digitais	Barramento bidirecional de dados
9	VCC	Alimentação	5Vcc
10	GND	Alimentação	0Vcc
11 a 14	A3 a A0	Entrada digital	Entrada de endereço para registradores internos
15	OEB	Entrada digital	Habilita saída de dados (8 bits)
16	WEB	Entrada digital	Habilita escrita (registradores internos)
17	CSB	Entrada digital	Habilita dispositivo (chip enable)
18	HREF	Saída digital	Saída de sincronismo horizontal
19	PCLK	Saída digital	Saída para sincronismo pixel a pixel
20	VSYNC	Saída digital	Saída de sincronismo vertical

Adaptado de Comedia (1999).

O módulo M4088, na verdade, é uma placa de circuito impresso composta pelo sensor OV5017, uma lente para concentrar a luz no sensor, alguns componentes discretos externos e dois conectores de saída: uma analógica (vídeo composto) e uma digital. Segundo Omnivision Technologies (1998), o OV5017 é um *chip* completo para câmera de vídeo analógica. Com taxa de exibição de *frames* ou de *pixels* programável para adaptar-se às necessidades dos sistemas externos, tem resolução de 8 bits e fornece *frames* de até 384 x 288 *pixels* (fig. 9). O conversor interno A/D pode converter o sinal de vídeo a 50fps (*frames per second*, ou quadros por segundo), e a conversão é sincronizada conforme taxa de exibição de *pixels*. Ele fornece sinais de referência padrão de tempo, como VSYNC, HREF, PCLK (vistos na tabela 2).

FIGURA 9 – DIVISÃO DA JANELA (*FRAME*) NO M4088



Fonte: Omnivision Technologies (1998).

A imagem é dividida em blocos (16 x 16), e cada bloco com 24 *pixels* de altura por 18 de largura. Usam-se os registradores internos para definir o tamanho da janela, definindo-se o endereço dos blocos inicial e final de cada direção. As características de definição de janela mudam somente o tempo de asserção de HREF, não influindo nas taxas de *pixel* ou dados. Se o endereço final for igual ou menor que o de início, então a janela vai do endereço de início até a borda mais a direita.

O controle de exposição é configurável para operação manual ou automática, e um divisor interno propicia diferentes taxas de exposição mesmo em operação, sem ter que mudar a freqüência de clock de entrada. Pode ainda operar em modo *single frame* (quadro único), que proporciona a transferência de um único *frame* pelo controle do sinal HREF.

2.3.1.1 BARRAMENTO DE DADOS

O barramento de dados é compartilhado negando-se OEB, e os dados de vídeo de 8 bits do conversor A/D são sincronizados com o PCLK. O nível mais baixo é 00h e o mais alto FFh, sem reserva de código para comandos. PCLK é o clock de *pixel* e pode ser contínuo ou presente somente durante *pixels* válidos. HREF é freqüentemente utilizado para qualificar o *pixel* pois permanece em nível alto (nível lógico 1) somente durante a região programada como janela válida.

Os dados de vídeo são atualizados no barramento de dados na borda de subida de PCLK e assegurados até a borda de descida. A leitura dos dados de vídeo não é diferente da leitura de um registrador interno, para isso requer-se a asserção de OEB e CSB além do endereço correto do registro de dados de vídeo. Para manter rajada ininterrupta de dados de vídeo, os dados de vídeo do OV5017 serão atualizados enquanto OEB, CSB e o endereço correto são mantidos nas respectivas entradas do módulo.

Para prevenir erros de *overrun* (sobrecarga de dados), deve-se ler pelo menos um dado de vídeo em cada período de clock de *pixel*. Os bits RDY e OV do registrador STATUS, visto no quadro 4, permitem ao *host* detectar eventuais erros.

2.3.1.2 REGISTRADORES DE CONTROLE

A leitura/escrita de registros internos é feita da mesma forma que a uma memória normal, usando-se os pinos DATA<7:0>, A<3:0>, WEB, OEB e CSB. O ciclo de leitura pode ser controlado pela seleção do *chip* ou por endereços. O ciclo de escrita também pode ser controlado pela seleção do *chip* ou por habilitação de escrita, e só afeta a registradores editáveis, não tendo nenhum efeito sobre registros de somente leitura, como o registrador de *status* de porta de vídeo. O ciclo de memória é totalmente assíncrono ao tempo de *frame* ou *pixel*.

Deve-se tomar cuidado com a alteração em registradores durante a transferência de dados de um *frame*, visto que a escrita em certos registros afeta a operação básica da câmera. A tabela 3 traz a descrição dos registradores internos. No anexo IV pode ser vista sua função e valor padrão.

TABELA 3 – REGISTRADORES INTERNOS DO M4088

Endereço	Mnemônico	Tipo	Função	Valor padrão
10xx	VPORT	R	Dados de vídeo	xxxxxxxx
0000	STATUS	R	Status	00xxxxxx
0001	FCTL	W	Controle fluxo frame único, controle do sistema	00xx0000
0010	EXCTL	R/W	Controle de exposição	11111111
0011	GCTL	R/W	Ganho	xxxxx000
0100	FRCTL	R/W	Divisor de taxa de frame	xx000000
0101	MCTL	R/W	Controles diversos	00000000
0110	HWCTL	R/W	Controle de janela	00000000
0111	VWCTL	R/W	Controle de janela	00000000
1110	TST	W	Reservado para teste	xxxxxxxx
1111	TOPT	W	Reservado para teste	xxxxxxxx
Nota: Os bits não implementados nos registros retornam 0 (zero) no ciclo de leitura e não são afetados no ciclo de escrita.				

Adaptado de Omnivision Technologies (1998).

2.4 VISUALIZAÇÃO DE IMAGENS

A visualização de imagens processadas eletronicamente envolve desde métodos de definição da cor até a compatibilização dos sistemas de coordenadas dos sistemas envolvidos. Todo esse esforço é desprendido a fim de gerar a imagem da maneira como foi capturada, ou se for o caso, de modo aproximado. Essas técnicas, além de outras abrangidas pela área de computação gráfica, são brevemente descritas em Corrigan (1994) e mais aprofundadas em Foley (1990).

O uso de *displays* de cristal líquido para visualização de imagens remonta a algumas décadas. Atualmente já são construídos televisores e monitores de microcomputadores usando essa tecnologia, graças ao nível de qualidade de imagem alcançado (Tech Data, 1999).

2.4.1 *DISPLAY* DE CRISTAL LÍQUIDO

Há dois tipos básicos de *displays*: os alfanuméricos (ou de segmentos) e os gráficos. Os alfanuméricos são encontrados em visores de calculadoras e relógios de pulso, enquanto os

gráficos estão presente em monitores de *notebooks* e telefones celulares. Mas existem outras formas de classificar esses dispositivos, principalmente:

a) quanto a sua cor: monocromáticos, tons de cinza e coloridos;
b) quanto ao tipo de iluminação: reflexivos (dependem da luz ambiente) e iluminados (possuem *backlight* - luz de fundo);
c) quanto à tecnologia de fabricação: passivos (comuns e mais baratos) e ativos (maior nitidez de imagem).

Segundo Oki Semiconductor (2002), os LCD´s são compostos por uma película de cristal líquido contida por duas lâminas de vidro que possuem trilhas condutivas para polarizar os cristais. Em seu estado normal (desenergizados) os cristais permitem que a luz atravesse diretamente, deixando-se ver a cor natural de fundo. Uma vez energizados, redirecionam a luz, que é absorvida causando a impressão de ocorrência de sombra.

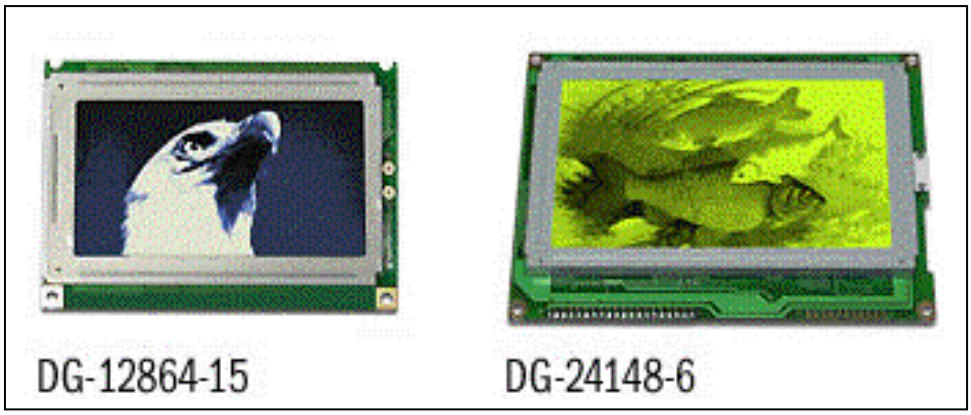
O sinal aplicado para polarizar as moléculas de cristal deve ser alternado, para evitar degradação da qualidade do *display*. Isto é feito a intervalos regulares, afim de executar o *refresh* da imagem, isto é, a imagem deve ser periodicamente reescrita, caso contrário ela gradativamente desaparece. Essa tarefa cabe a controladores dedicados, ou *lcd controllers*, embora possam existir *displays* sem esses dispositivos (Hantronix, 2000) (Seiko, 1999).

2.4.2 MÓDULOS LCD

Normalmente denominam-se módulos LCD os conjuntos formados pela tela de cristal líquido, dispositivos para reflexão da luz ou iluminação e demais componentes eletrônicos, com ou sem controladores (Oki Semiconductor, 2002) – conforme vistos na fig. 10.

Os módulos que contém controladores são mais práticos, pois podem ser configurados através de registros internos, além de evitar o trabalho de implementar a varredura. A escrita e leitura desses registros, de forma análoga aos módulos de câmera digital, também é feita como se o dispositivo fosse uma memória ou outro circuito digital endereçável. Data International (2002) e Excel Tecnology (2002b) apresentam várias opções de módulos LCD, sendo que este último disponibiliza *datasheets* sobre diversos deles.

FIGURA 10 – MÓDULOS LCD



Fonte: Data International (2002).

2.4.3 *DISPLAY* GRÁFICO

A tarefa de mostrar as imagens capturadas pela câmera digital fica então a cargo de um módulo LCD. Por questão de disponibilidade, dois módulos foram estudados, o HDM240200 e o DG16080. O fabricante (Hantronix, 1998) disponibiliza documentação sobre o primeiro, um *display* gráfico monocromático de 240 colunas por 200 linhas que, entretanto, não contém o controlador para simplificar a operação de escrita.

O módulo DG-16080, por outro lado, possui controlador LC7981 da Sanyo (2002), e tem sua pinagem descrita na tabela 4. Trata-se também de um *display* gráfico monocromático de 160 colunas por 80 linhas com *backlight*.

TABELA 4 – DESCRIÇÃO DOS PINOS DO MÓDULO LCD DG-16080

Pino	Mnemônico	Tipo	Função
1	Vss	Alimentação	0Vcc
2	Vdd	Alimentação	Alimentação dos circuitos digitais
3	Vo	Alimentação	Alimentação do display
4	RS	Entrada digital	Seleciona instrução (1) / dado (0)
5	R/W	Entrada digital	Seleciona leitura (1) / escrita (0)
6	E	Entrada digital	Habilita operação (enable)
7 a 14	DB0 a DB7	Entrada/saída digitais	Barramento bidirecional de dados
15	CS	Entrada digital	Habilita dispositivo (chip enable)
16	Res	Entrada digital	Reset
17	Vee	Alimentação	Saída de voltagem (-10V)
18 a 20	NC	-	Não conectados

Adaptado de Excel Tecnology (2002a).

A escrita é feita da esquerda para a direita, de cima para baixo, de oito em oito *pixels*. Apesar de ser possível configurá-lo também para trabalhar com palavra de 6 ou 7 bits. Outras configurações também são necessárias, entre elas:

a) modo de trabalho: gráfico ou caractere;
b) número de bytes ou caracteres por linha;
c) número de amostragens (varredura);
d) posição do cursor.

O método de configuração do módulo DG-16080 é semelhante ao usado para programar o módulo de câmera digital. Porém há somente um barramento que serve como endereçamento e dados. O procedimento para escrita no módulo é feito em duas etapas. Primeiro passa-se o endereço do registrador interno e depois o dado a ser escrito nele. Cada etapa é executada da seguinte forma:

a) joga-se o byte no barramento;
b) seleciona-se o tipo de operação (escrita/leitura) no pino R/W (pino 5);
c) através do pino RS (pino 4) é definido se o byte presente no barramento é um endereço ou dado;
d) pulsa-se o pino E (pino 6) para habilitar a operação.

O ciclo de leitura é semelhante. Na primeira etapa escreve-se o endereço do registrador a ser lido e, para executar a segunda etapa, basta aplicar nível baixo (nível lógico 0) no pino R/W. Logo após ter sido habilitada a operação, o dado está disponível no barramento para ser lido.

Em qualquer um dos casos, escrita ou leitura, antes de qualquer etapa, deve-se verificar um *flag* interno do módulo para assegurar que a operação será bem sucedida. Isto é feito executando-se um ciclo de leitura com o pino RS em nível alto. O registrador interno *busy flag* pode ser lido no bit mais significativo do dado lido.

3 FERRAMENTAS UTILIZADAS

Este capítulo é especialmente dedicado às ferramentas que auxiliaram na especificação e no desenvolvimento do protótipo, sejam elas ferramentas de software ou de hardware. A intenção é tentar transcrever algumas minúcias que envolveram o projeto e a implementação, no intuito de que as experiências possam ser recriadas a contento.

Portanto, recordando o disposto na parte introdutória do capítulo 2 (fundamentação teórica), serão descritas algumas ferramentas utilizadas que remontam à formação básica em eletrônica e que merecem ser mencionadas.

3.1 FERRAMENTA DE ESPECIFICAÇÃO DE SOFTWARE

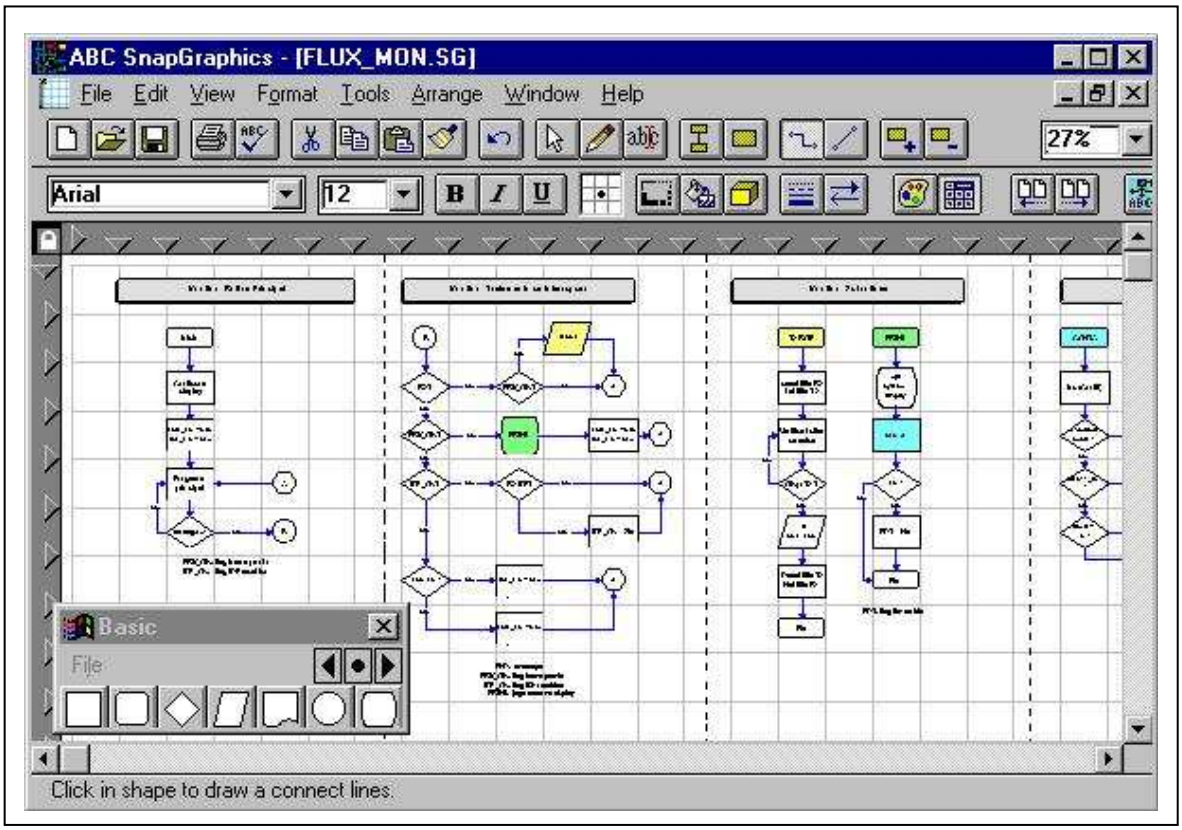
Toda a especificação foi feita utilizando-se de fluxogramas para representar o problema, já que o código obedece basicamente um andamento seqüencial. Mesmo quando foi necessário especificar partes mais complexas como interrupções geradas no microcontrolador, uma solução adequada foi encontrada através da fluxogramação.

Esta técnica foi escolhida pela familiaridade, e é muito eficiente se for bem utilizada, sendo bem aceita por diferentes áreas do conhecimento. Muitos manuais e catálogos de componentes eletrônicos apresentam diagramas nesse estilo.

A ferramenta utilizada para a edição dos fluxogramas foi o ABC SnapGraphics versão 2.0a da Micrografx Inc, visto na fig. 11. Ela faz parte de um pacote de aplicativos especializados na edição de diagramas, sendo o ABC FlowCharter o mais conhecido.

O aplicativo foi escolhido por sua simplicidade e disponibilidade, prestando-se muito bem ao que foi destinado. O ABC SnapGraphics ainda auxiliou na criação de outros gráficos utilizados no trabalho.

FIGURA 11 – TELA DO ABC SNAPGRAPHICS

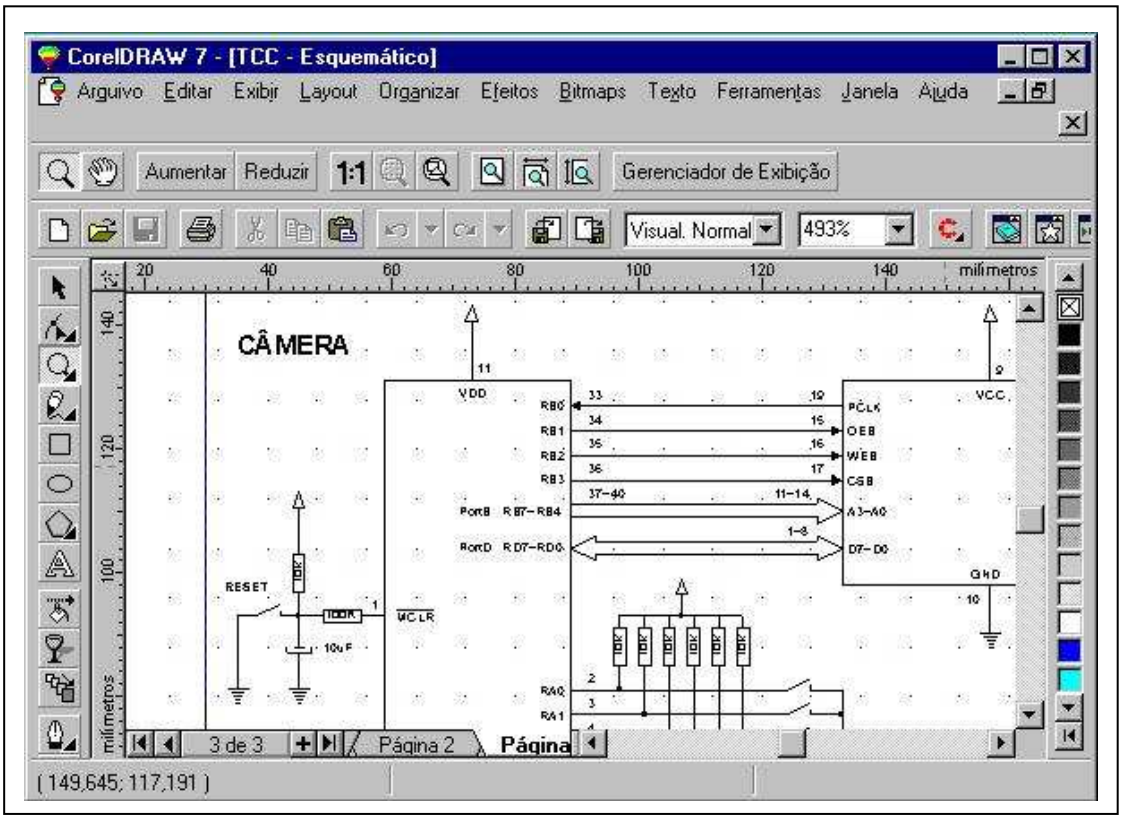


3.2 FERRAMENTA DE ESPECIFICAÇÃO DE HARDWARE

A edição de esquemáticos ficou a cargo do CorelDraw versão 7, da Corel Corporation, embora não seja um software destinado a essa aplicação. Vários editores de esquemas elétrico/eletrônicos foram estudados e não havia nenhum disponível e gratuito que contivesse as bibliotecas necessárias ao desenvolvimento do protótipo.

Entretanto, como pode ser visto na fig. 12, o circuito é bastante simples e o protótipo pôde ser montado sem a necessidade de confeccionar-se uma placa específica para tal, o CorelDraw passou a ser a alternativa mais viável – embora não específica para a tarefa.

FIGURA 12 – TELA DO CORELDRAW



3.3 FERRAMENTAS DE IMPLEMENTAÇÃO

As ferramentas de implementação de hardware e de implementação de software acabam muitas vezes se fundindo num mesmo assunto. Serão, portanto, apresentadas em subtítulos deste capítulo, buscando-se dispô-las por ordem de importância.

Serão descritos primeiramente o aplicativo de desenvolvimento de software para microcontroladores e o gravador universal de dispositivos, exaustivamente utilizado para gravar o programa nos microcontroladores durante a fase de desenvolvimento.

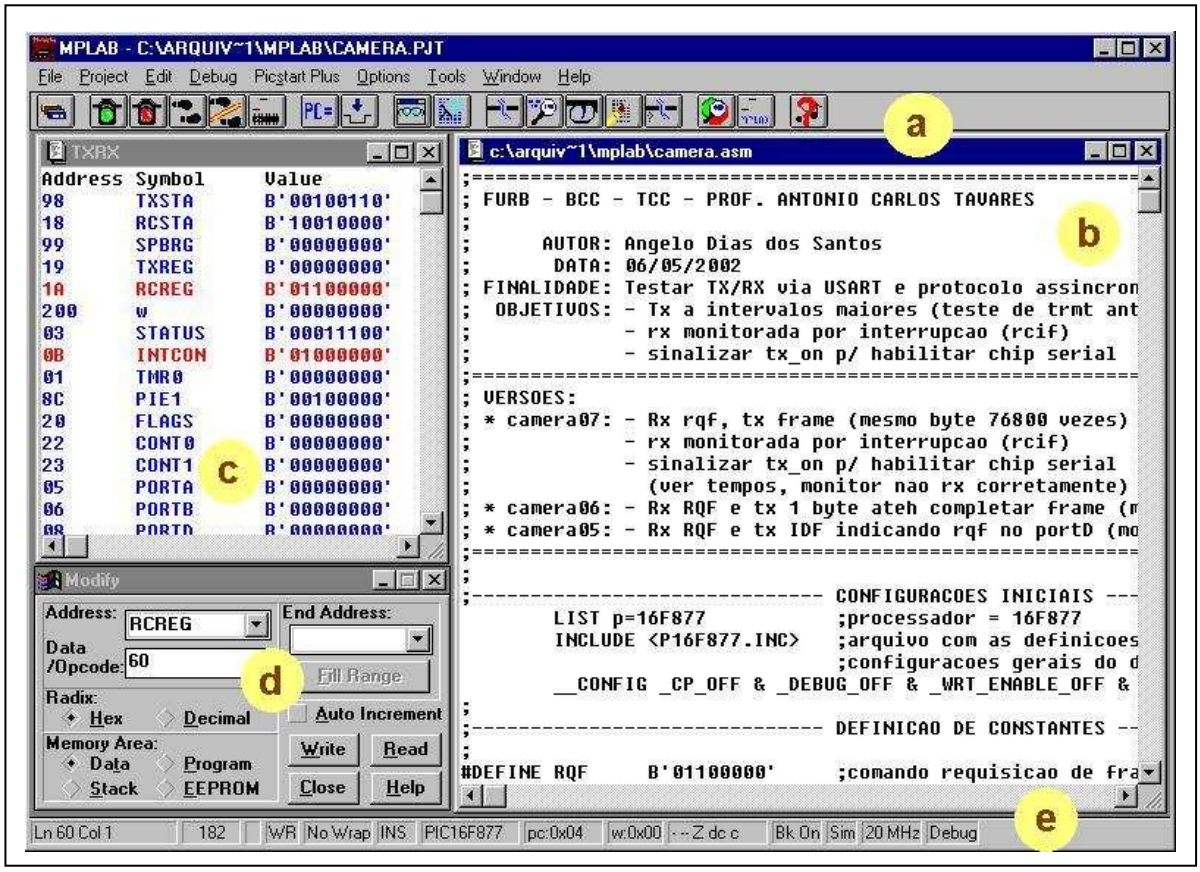
Em seguida, serão descritas outras ferramentas não usuais que também foram utilizadas no decorrer da pesquisa e da implementação do protótipo. Fala-se em pesquisa pois muitas experiências foram realizadas até chegar-se ao resultado final, e estes equipamentos desempenharam papel fundamental para que se atingisse o objetivo. São eles o *protoboard*, o osciloscópio e o multímetro digital – todos conhecidos tanto por profissionais quanto por aficcionados em eletrônica.

3.3.1 AMBIENTE DE DESENVOLVIMENTO

Conforme visto no capítulo 2.1.3.4 (considerações sobre programação), o ambiente de desenvolvimento MPLab é gratuitamente fornecido pela Microchip (2000) e é considerado um dos motivos pela popularização dos microcontroladores da família PIC (Souza, 2000), o que representa um bom motivo para sua adoção.

O processo de desenvolvimento no MPLab começa pela criação de um projeto, onde são vinculados o programa fonte a informações sobre o microcontrolador utilizado, além de configurações de janelas auxiliares e dos processos de simulação/emulação. A fig. 13 demonstra a tela do MPLab flagrada durante o desenvolvimento do protótipo.

FIGURA 13 – TELA DO MPLAB



Assinalados na fig. 13 estão alguns itens que compõe o aplicativo e que serão brevemente descritos a seguir:

a) barra de ferramentas: são quatro barras cambiadas pelo botão mais à esquerda e que oferecem auxílio desde a edição até a simulação do projeto;

b) programa fonte: a linguagem MPAsm define características específicas a cada microcontrolador em arquivos fornecidos junto com o MPLab e que são acrescentadas ao projeto através da instrução *include* – o modelo do microcontrolador pode ser visto no canto inferior esquerdo;
c) visualização de registros: as alterações são grifadas em vermelho, tanto nos SFR's quanto nos registros de uso geral, definidos pelo programador;
d) janela de alteração: nela podem ser alterados os valores de todos os registros pelo nome ou por posição de memória, para facilitar a simulação;
e) barra de *status*: informações gerais sobre o ambiente.

As facilidades não param por aí, pois vão desde visualização do conteúdo da pilha até janelas com botões configuráveis para estímulos externos a pinos de entrada do microcontrolador. A maior parte das opções de janelas são devidamente salvas com as configurações no arquivo do projeto (<*nome_do_arquivo*>.pjt).

A compilação e tradução para código de máquina também são feitas diretamente no MPLab e o resultado é mostrado numa janela com indicação da linha que contém o eventual erro ou as advertências corriqueiras (*warnings*). Depois de corretamente compilado, um arquivo em formato hexadecimal é gerado (<*nome_do_arquivo*>.hex) contendo o código de máquina para ser gravado no dispositivo. A partir deste arquivo gerado, o projeto pode ser executado no microcomputador de três maneiras:

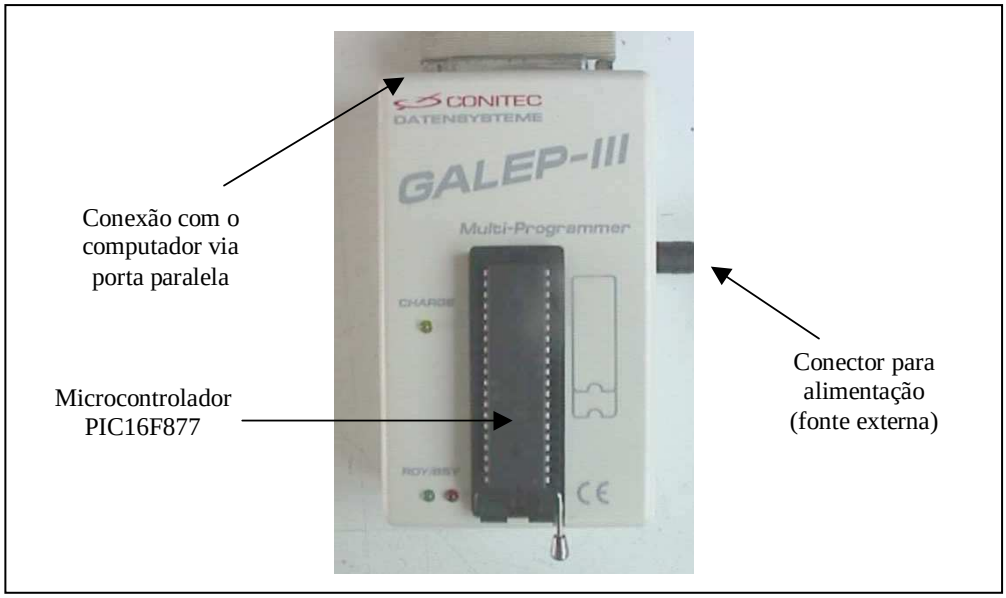
a) *run* (executar): o programa roda direto e a interface não é atualizada até alguma interrupção – como a passagem por um *breakpoint* ou a execução de um *halt*;
b) *animate* (animação): a interface é atualizada conforme o programa vai sendo executado (torna-se mais lento por isso);
c) *step* (passo-a-passo): a interface é atualizada a cada novo passo.

A linguagem é bastante enxuta e confunde-se com a ferramenta (são as 35 instruções já mencionadas e outras poucas diretivas de compilação). Existem outras opções de compiladores menos interativos, baseados em DOS, que todavia não foram testados.

3.3.2 GRAVADOR UNIVERSAL DE DISPOSITIVOS

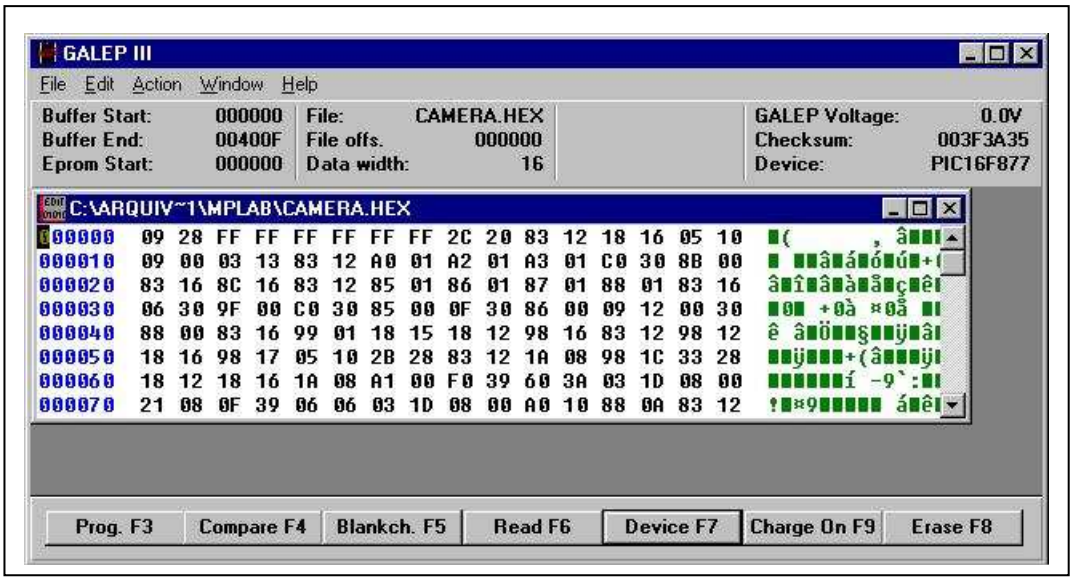
O gravador usado durante todo o desenvolvimento foi o Galep-III da Conitec, visto na fig. 14. Compatível com a família de microcontroladores PIC, pode ser usado para gravar outros componentes como eprom´s, eeprom´s, ram´s, além de outras famílias de microcontroladores de diversos fabricantes. O aparelho ainda possui bateria recarregável, o que permite o uso com microcomputadores portáteis.

FIGURA 14 – GALEP-III: HARDWARE



Há uma grande variedade de gravadores universais no mercado, e o cuidado que se deve ter para sua escolha diz respeito justamente aos tipos de dispositivos compatíveis que se quer gravar. Além disso, o software que acompanha o gravador também deve ser analisado. O Galep-III vem acompanhado por software de mesmo nome, para ambiente Windows. Na fig. 15 pode-se ver detalhes de sua interface.

FIGURA 15 – GALEP-III: SOFTWARE



Na parte inferior da janela do programa vê-se sete botões, ativados pelo *mouse* ou pelas respectivas teclas de função, e que dão acesso à maioria das opções e ações do gravador:

a) F3 – grava o dispositivo;
b) F4 – lê dados do dispositivo que está no gravador e compara com o arquivo carregado na memória do programa de gravação;
c) F5 – verifica se o dispositivo está apagado;
d) F6 – lê dados do dispositivo que está no gravador e carrega na memória;
e) F7 – configuração de tipo de dispositivo;
f) F8 – apaga dispositivo;
g) F9 – habilita carga da bateria interna do Galep-III.

Após a geração do arquivo no formato desejado, no caso o formato hexadecimal da Intel (.hex) gerado pelo MPLab a partir do programa fonte, procede-se à leitura deste pelo software do Galep-III (menu *File*, opção *Load File...*). Depois disso, com o software devidamente configurado para o dispositivo utilizado e com o dispositivo fisicamente posicionado no gravador conforme fig. 13, passa-se à gravação do mesmo. Finda a gravação o equipamento ainda faz uma verificação, através de *checksum* gerado com base no arquivo carregado na memória, para detectar possíveis falhas.

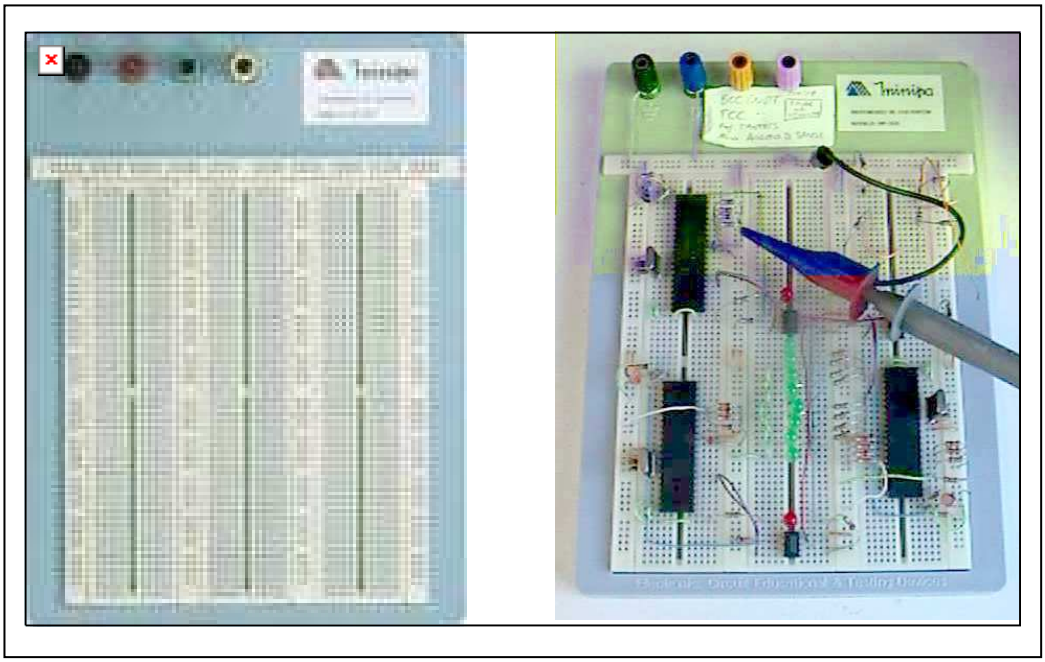
3.3.3 PROTOBOARD

Para efetuar a montagem do circuito eletrônico foi utilizado o *protoboard*. Do inglês *prototype board* ou placa de protótipo, é uma ferramenta prática e atualmente bastante acessível, usada para construir circuitos temporários e realizar experiências eletrônicas (Doctronics, 2002) (Minipa, 2002).

O equipamento consiste num conjunto de placas contendo fileiras de contatos elétricos, distribuídos de forma a facilitar a montagem de circuitos eletrônicos pelo simples encaixe de fios e componentes, sem a necessidade de soldagem. Isto faz com que os componentes possam ser reaproveitados, além de possibilitar rápidas alterações nas ligações elétricas.

Segundo Minipa (2002), o corpo do *protoboard* normalmente é feito de polímero ABS (isolante elétrico) e os contatos compostos por uma liga de prata e níquel. A fig. 16 mostra o *protoboard* utilizado, o modelo MP-2420 da Minipa, em dois momentos: sem componentes e durante testes com parte do circuito do protótipo.

FIGURA 16 – PROTOBOARD



Existem alguns inconvenientes, porém, em relação ao uso de *protoboards*. Depois de algum tempo de uso a pressão exercida pelos contatos elétricos tende a diminuir provocando problemas de condução elétrica. Além disso, capacitâncias parasitas características deste tipo

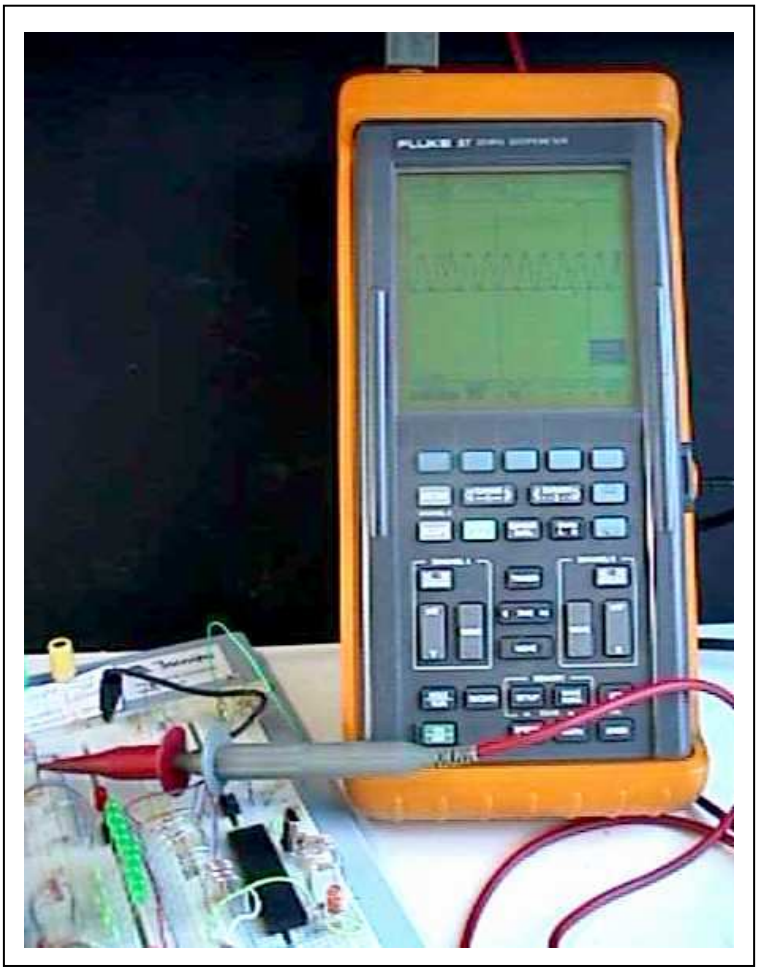
de equipamento podem alterar o comportamento elétrico do circuito, podendo também causar falhas, difíceis de serem diagnosticadas.

3.3.4 OSCILOSCÓPIO

É um instrumento de bancada que permite a visualização de formas de onda, ou seja, as oscilações dos sinais elétricos na unidade de tempo. Possui pelo menos dois canais nos quais podem ser capturados sinais elétricos que podem ser visualizados e manipulados de forma independente. Sua principal característica é a freqüência de amostragem do sinal, que chega a até 500MHz em modelos digitais mais atuais (Tektronix, 2002).

Na fig. 17 pose ser visto um dos modelos utilizados durante a implementação: o Fluke 97, um osciloscópio digital de 60MHz, portátil, com tela de cristal líquido.

FIGURA 17 – OSCILOSCÓPIO DIGITAL PORTÁTIL



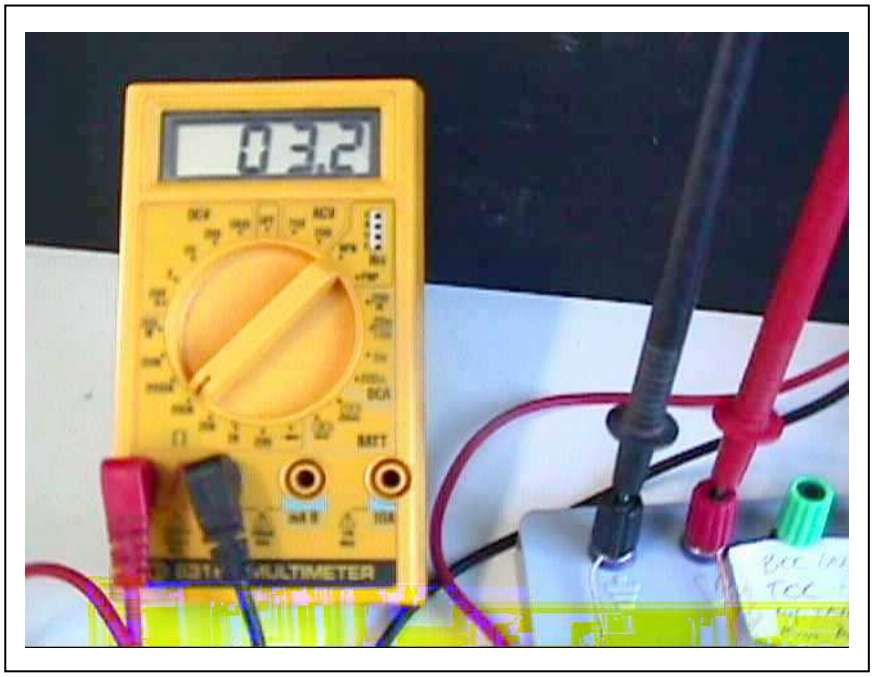
Os sinais elétricos são visualizados em uma tela que possui um plano cartesiano, onde a tensão é representada no eixo das coordenadas e o tempo no eixo das abcissas. O osciloscópio permite a rápida verificação de eventos, como alterações nas saídas do microcontrolador, sem a necessidade de componentes como *led´s* para visualização. Além disso, certos eventos que ocorrem em intervalos de tempo muito pequenos também podem ser observados na tela – para modelos de 20MHz pode-se flagrar sinais de até 50ns (0,00000005 segundos) de duração.

Outros modelos utilizados foram o TDS200 (100MHz/digital) da Tektronix, o GOS-620 da Instek e o MO-1222 da Minipa, ambos de 20MHz.

3.3.5 MULTÍMETRO DIGITAL

Multímetro, como o nome sugere, por ser um equipamento capaz de realizar medições de diversas grandezas elétricas, e digital por causa da tecnologia empregada em sua fabricação. A fig. 18 mostra um dos modelos utilizados durante o desenvolvimento.

FIGURA 18 – MULTÍMETRO DIGITAL



Também chamado de *multiteste*, normalmente é portátil e reúne ohmímetro, amperímetro e voltímetro num único aparelho. Atualmente porém é difícil encontrar um

multímetro que não inclua testador de continuidade, testador de diodos e medidor de ganho de transistores. Os mais sofisticados ainda podem conter freqüencímetro, capacímetro, medidor de níveis lógicos e até termômetro. A seguir encontra-se um resumo das partes do multímetro utilizadas no desenvolvimento:

a) ohmímetro: mede a resistência de condutores à passagem da corrente elétrica, que tem o *ohm* (Ω) como unidade e é normalmente chamada de resistência – foi utilizada para conferir valores de resistores usados no protótipo;
b) amperímetro: mede a intensidade da corrente elétrica passando por um condutor, que tem o ampère (A) como unidade e é normalmente chamada de corrente;
c) voltímetro: mede a diferença de potencial entre dois pontos, normalmente chamada tensão, que tem o volt (V) como unidade – foi usado para verificação de possíveis falhas na ligação ou até mesmo na programação dos microcontroladores, o que poderia acarretar um curto-circuito levando à queda da tensão;
d) testador de continuidade: testa se há condução elétrica entre dois pontos, normalmente emitindo um beep quando se houver condutividade e se a resistência elétrica não ultrapassar 100 ohms – usado para conferência de ligações elétricas e do estado de cabos usados no desenvolvimento;
e) testador de diodos: usado para testar *led´s* que são diodos emissores de luz - pequenas luzes encontradas em equipamentos eletrônicos, normalmente nas cores verde ou vermelha);
f) capacímetro: mede a capacidade de armazenar cargas elétricas, ou capacitância, que tem o farad (F) como unidade – foi utilizado para conferir valores de capacitores utilizados no protótipo.

4 DESENVOLVIMENTO DO TRABALHO

O trabalho foi concebido de forma modular, onde cada uma de suas três partes foi especificada, implementada, testada e validada para depois ser incorporada ao conjunto final. Começando pela rede, passando pela implementação da visualização da imagem e somente por último, a captura da mesma.

Mas há uma lógica por trás desse planejamento: o motivo da escolha da rede para iniciar o desenvolvimento do protótipo é simples. A comunicação de dados é o ponto mais crítico, pois imagens envolvem grandes quantidades de dados. Se a rede não suportar a transferência de ao menos um *frame* por segundo, o monitoramento não será eficiente, perdendo o sentido da aplicação do protótipo.

Uma vez implementada a rede, com protocolo e interligações definidas, o próximo passo é a visualização da imagem. O motivo também é simples: é mais fácil testar a implementação do monitor do que a implementação da câmera. Basta fazer com que uma câmera imaginária, implementada somente para enviar dados, envie imagens de controle para teste do *display*.

Com o monitor funcional, pode-se então implementar a aquisição da imagem, fechando o ciclo de desenvolvimento. A cada etapa implementada são feitos testes e validação do protótipo, já que se uma parte não estiver corretamente depurada, a andamento do trabalho certamente será afetado. O ciclo de vida é, na verdade, bastante intuitivo.

O trabalho foi totalmente desenvolvido em plataformas Pentium (IBM PC compatíveis), com sistema operacional Windows 9x, sendo o protótipo implementado em linguagem *assembly* do microcontrolador PIC16F877.

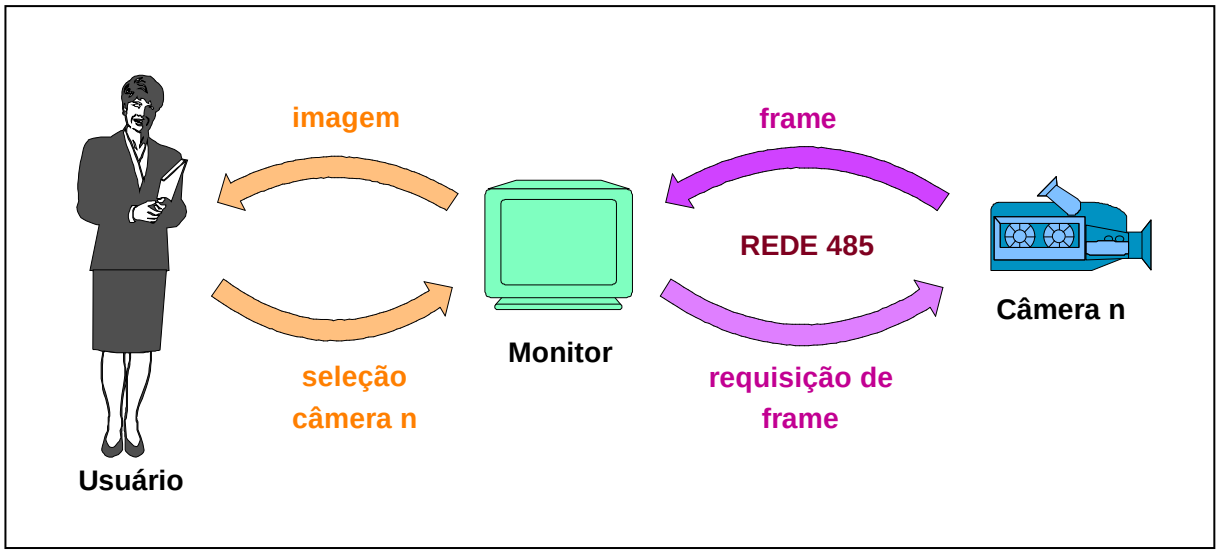
4.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O protótipo deve permitir que se visualize, em um monitor de cristal líquido, imagens capturadas por câmeras digitais remotamente localizadas, através de uma rede de comunicação de dados baseada em par trançado.

As distâncias envolvidas vão desde poucos centímetros até cerca de 100 metros entre os nós de rede mais afastados, devido à degradação do sinal e às taxas de transmissão requeridas. A rede deve prever a conexão de até 32 pontos, entre câmeras e monitores. A taxa de amostragem deve ser da ordem de uma imagem por segundo, tornando o trabalho passível de ser utilizado para monitoramento de ambientes.

Resumidamente, o usuário deve selecionar o número (endereço) da câmera que deseja visualizar. O monitor, enquanto ligado, envia à câmera requisições de *frame*, a intervalos regulares. A câmera cujo endereço conferir com o da solicitação, responde enviando um *frame* ao monitor, que o joga na tela. A fig. 19 mostra o diagrama funcional (um diagrama de contexto adaptado às circunstâncias) representando graficamente o protótipo.

FIGURA 19 – DIAGRAMA FUNCIONAL DO PROTÓTIPO



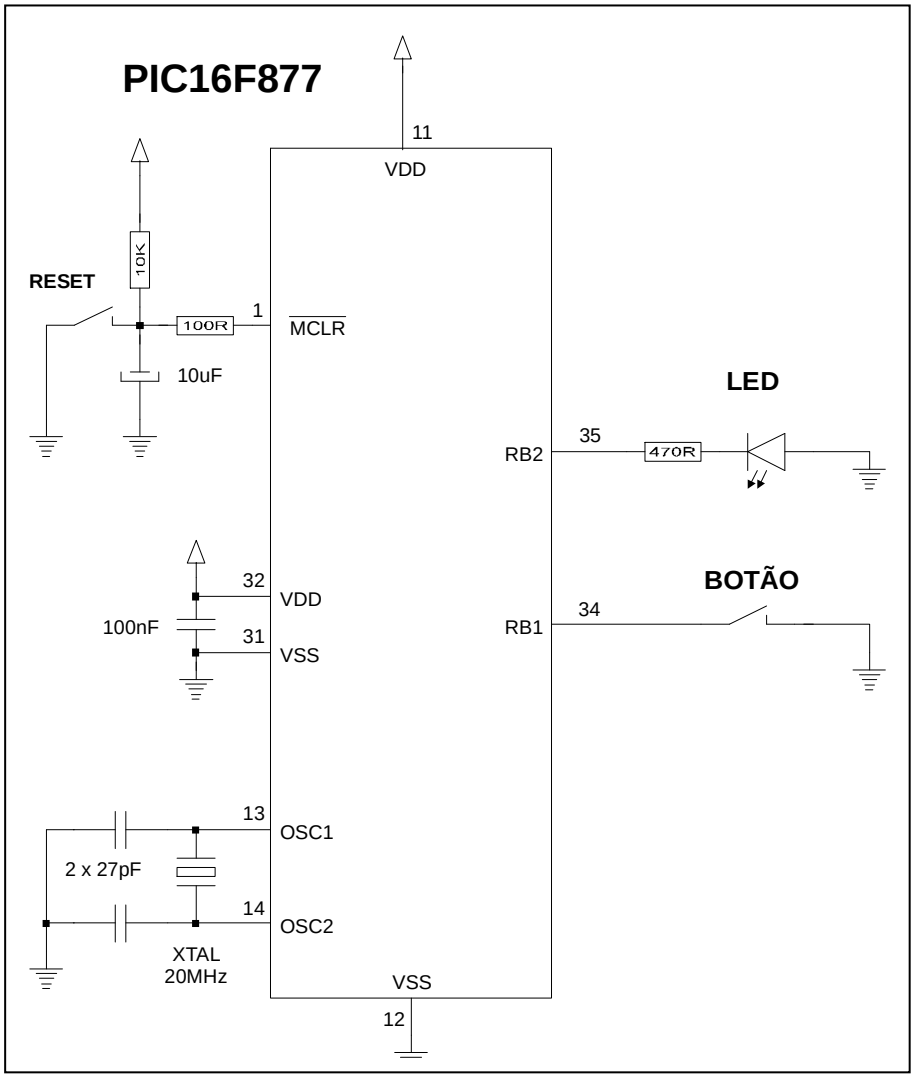
4.2 CONSIDERAÇÕES INICIAIS SOBRE HARDWARE E SOFTWARE

O primeiro passo em relação ao desenvolvimento do trabalho é conhecer bem o microcontrolador definido para o protótipo. Para isso, sugere-se uma pequena implementação a fim de adquirir familiarização tanto com o hardware quanto com o software que serão empregados.

Como teste inicial de codificação, sugerido por Souza (2000), foi implementado o conhecido “botão e *led*”: o estado de um botão é monitorado por um *led*. Quando o botão é acionado, acende um *led* ligado a uma saída do microcontrolador.

Souza (2000), Silva Júnior (1997) e Microchip (2001) auxiliam na tarefa de determinar o circuito mínimo para que o microcontrolador possa funcionar adequadamente. A fig. 20 mostra, então, o diagrama elétrico utilizado para esta implementação experimental, enquanto o código pode ser visto no quadro 2.

FIGURA 20 – DIAGRAMA ELÉTRICO BOTÃO E *LED*



QUADRO 2 – PROGRAMA FONTE BOTÃO E *LED*

```
;*****************************************************************
; FURB - BCC/NOT  TCC        ANTONIO C. TAVARES
; ALUNO: Ângelo Dias dos Santos
; Este programa representa o estado de um botao atraves de um led
;*****************************************************************
; CONFIGURACOES INICIAIS
;*****************************************************************
            LIST   p = PIC16F877            ;seleciona tipo de  microprocessador
            #INCLUDE <P16F877.INC>      ;arquivo de definicao P/ 16F877
;           configura power on reset timer on, watchdog off e xtal HS:
            __CONFIG _PWRTE_ON & _WDT_OFF & _HS_OSC
;*****************************************************************
; ENTRADAS
;*****************************************************************
#DEFINE BOTAO    PORTB,1  ;porta do botao (RB1 - pino 34)
                                                        ;0 -> pressionado
                                                        ;1 -> liberado
;*****************************************************************
; SAÍDAS
;*****************************************************************
#DEFINE LED         PORTB,2             ;porta do led (RB2 - pino 35)
                                                        ; 0 -> apagado
                                                        ; 1 -> aceso
;*****************************************************************
; VETOR DE RESET
;*****************************************************************
            ORG         0x00                                ;endereco inicial
            GOTO        INICIO
;*****************************************************************
; VETOR DE INTERRUPCAO
;*****************************************************************
            ORG         0x04                                ;endereco    inicial
interrupcao
            RETFIE                                  ;retorna interrupcao
;*****************************************************************
; CONFIGURACAO DO SISTEMA
;*****************************************************************
INICIO
            BCF                 STATUS,RP1
            BSF                 STATUS,RP0          ;seleciona banco 1
            MOVLW  B'00000010'          ;define rb1 como entrada e demais como
            MOVWF  TRISB                ;...saídas
            BCF                 OPTION_REG,7        ;pull-ups habilitados em rb
            CLRF        INTCON              ;todas interrupcoes desligadas
            BCF                 STATUS,RP0          ;seleciona banco 0
;*****************************************************************
; INICIALIZACAO DE VARIAVEIS
;*****************************************************************
            CLRF        PORTA               ;limpa PORTA
            CLRF        PORTB               ;limpa PORTB
            CLRF        PORTC               ;limpa PORTC
            CLRF        PORTD               ;limpa PORTD
            CLRF        PORTE               ;limpa PORTE
;*****************************************************************
;  ROTINA PRINCIPAL
;*****************************************************************
MAIN
            BTFSS       BOTAO               ;botao pressionado?
            GOTO        BOTAO_LIB           ;nao, entao trata botao liberado
            GOTO        BOTAO_PRES          ;sim, entao trata botao pressionado
BOTAO_LIB
            BCF                 LED                             ;apaga led
            GOTO        MAIN                ;retorna ao programa principal
BOTAO_PRES
            BSF                 LED                             ;acende led
            GOTO        MAIN                ;retorna ao programa principal
;*****************************************************************
; FIM DO PROGRAMA
;*****************************************************************
            END                             ;obrigatorio
```

O código foi incluído para que se possa ter uma visão da estruturação do programa, adaptado de uma sugestão também apresentada em Souza (2000). Retirando-se as observações e demais caracteres, o programa ficaria conforme visto no quadro 3.

QUADRO 3 – PROGRAMA FONTE BOTÃO E *LED* (RESUMIDO)

```
LIST   p = PIC16F877
           #INCLUDE <P16F877.INC>
           __CONFIG _PWRTE_ON & _WDT_OFF & _HS_OSC
#DEFINE BOTAO   PORTB,1
#DEFINE LED     PORTB,2
           ORG     0x00
           GOTO    INICIO
           ORG     0x04
           RETFIE
INICIO
           BCF     STATUS,RP1
           BSF     STATUS,RP0
           MOVLW   B'00000010'
           MOVWF   TRISB
           BCF     OPTION_REG,7
           CLRF    INTCON
           BCF     STATUS,RP0
           CLRF    PORTA
           CLRF    PORTB
           CLRF    PORTC
           CLRF    PORTD
           CLRF    PORTE
MAIN
           BTFSS   BOTAO
           GOTO    BOTAO_LIB
           GOTO    BOTAO_PRES
BOTAO_LIB
           BCF     LED
           GOTO    MAIN
BOTAO_PRES
           BSF     LED
           GOTO    MAIN
           END
```

As diretivas de compilação no início do arquivo, na parte de configurações iniciais, indicam ao compilador desde o tipo do microprocessador até o tipo de cristal oscilador usado. Souza (2000), Silva Júnior (1997) e sobretudo Microchip (2001), trazem explicações mais detalhadas das cláusulas e diretivas de compilação.

É importante salientar, também, que o circuito elétrico foi alterado diversas vezes durante o desenvolvimento do protótipo, já que a interface DS3696, o *display* e a câmera necessitam sinais de controle. Para que o trabalho fique mais didático e reduzido, será apresentada a disposição final de entradas e saídas do microcontrolador em todos os circuitos demonstrados daqui por diante.

4.3 REDE

A implementação do protótipo teve início, como já visto, pela especificação e desenvolvimento da rede de comunicação de dados que fará o compartilhamento da imagem. Esta é a parte principal e mais longa da implementação de software do protótipo, por isso será a primeira a ser descrita, seguida pela implementação do *display* e logo após, a anexação da câmera digital.

4.3.1 ESPECIFICAÇÃO

Para iniciar a especificação da rede, é necessário determinar o tipo de dados que vai ser transmitido. Como toda a implementação gira em torno da imagem (aquisição, compartilhamento, visualização), partiu-se primeiro para a definição do tamanho do *frame*.

Inicialmente foi previsto *frame* de 320 por 240 *pixels*, totalizando 76800 *pixels* ou 75Kbytes (um byte para cada *pixel*). Este formato foi baseado em tamanhos padrão de *displays* existentes, os quais poderiam ser utilizados no protótipo.

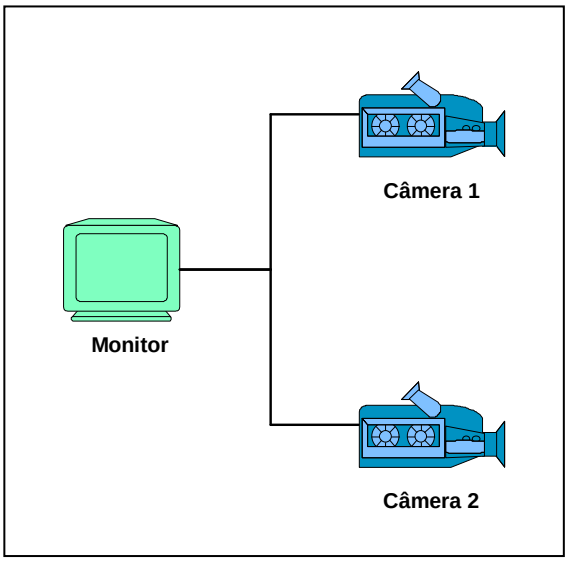
Como a topologia e o modo de transmissão estão pré definidos pelas características do protótipo, falta especificar o protocolo e determinar o circuito de interface entre o microcontrolador e a rede de comunicação.

4.3.1.1 TOPOLOGIA

Para facilitar o entendimento, a topologia do protótipo é recapitulada na fig. 21. Os termos *Monitor* e *Câmera* passarão a designar todo o conjunto de hardware e software que eles representam.

Este arranjo deve-se à disponibilidade de componentes necessários à confecção do protótipo. Ao todo foram usados três microcontroladores PIC16F877, um *display* DG-16080 e duas câmeras M4088. Os valores dos componentes discretos podem ser vistos nos próprios diagramas elétricos apresentados.

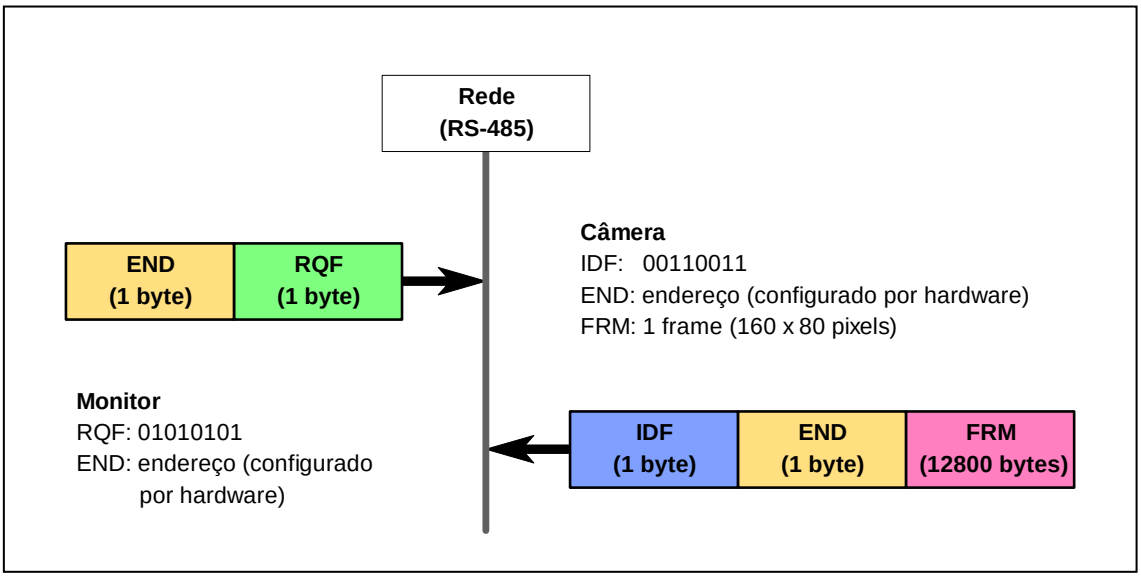
FIGURA 21 – TOPOLOGIA DO PROTÓTIPO



4.3.1.2 PROTOCOLO DE TRANSMISSÃO DE DADOS

Os estudos relativos à implementação da rede priorizaram a velocidade de transmissão, devido à característica do dado a ser transferido. Assim sendo, e considerando também as tecnologias envolvidas e a topologia proposta, foi sugerido protocolo proprietário (baseado em padrões assíncronos) a fim de tornar a transferência do *frame* o mais rápida possível. O protocolo está representado na fig. 22.

FIGURA 22 – PROTOCOLO DE TRANSMISSÃO



Foram escolhidos mnemônicos de fácil interpretação, buscando o uso de termos que facilmente lembrassem sua função. Assim:

a) RQF: requisição de *frame*;
b) IDF: início de *frame*;
c) END: endereço;
d) FRM: *frame*.

Os comando estão baseados num padrão de bits que pode ser facilmente visualizado no osciloscópio. O endereço é definido pelo hardware, mas de duas maneiras distintas:

a) no Monitor, que transmite a requisição de *frame*, o endereço é definido por chave acessível ao usuário, permitindo a rápida escolha da câmera a ser visualizada;
b) na Câmera, que transmite o *frame*, o endereço é único e determinado na instalação de cada unidade de captura de imagem.

Todos os dispositivos, ao serem inicializados, passam a ouvir a rede. Um *timer* em Monitor é acionado e, passados 13ms, na primeira oportunidade (rede livre para TX) Monitor enviará o comando RQF. Assim que enviar os dois bytes do comando (RQF + endereço da Câmera requisitada) o Monitor permanecerá ouvindo a rede até receber um comando IDF ou até que o temporizador estoure novamente (*timeout*), fechando o ciclo.

Vários Monitores configurados com um mesmo endereço poderão receber o mesmo frame enviado pela Câmera requisitada. Basta que um dos Monitores envie um comando RQF e que todos estejam aptos a receber o frame. Esta solução foi proposta para tornar a rede ainda mais rápida.

Câmera, por sua vez, fica sempre monitorando a rede até receber o comando RQF do Monitor. Todas Câmeras recebem o comando ao mesmo tempo, mas somente a que contiver o endereço correto é quem irá responder. A outra continua monitorando até receber um comando válido, isto é, RQF e endereço.

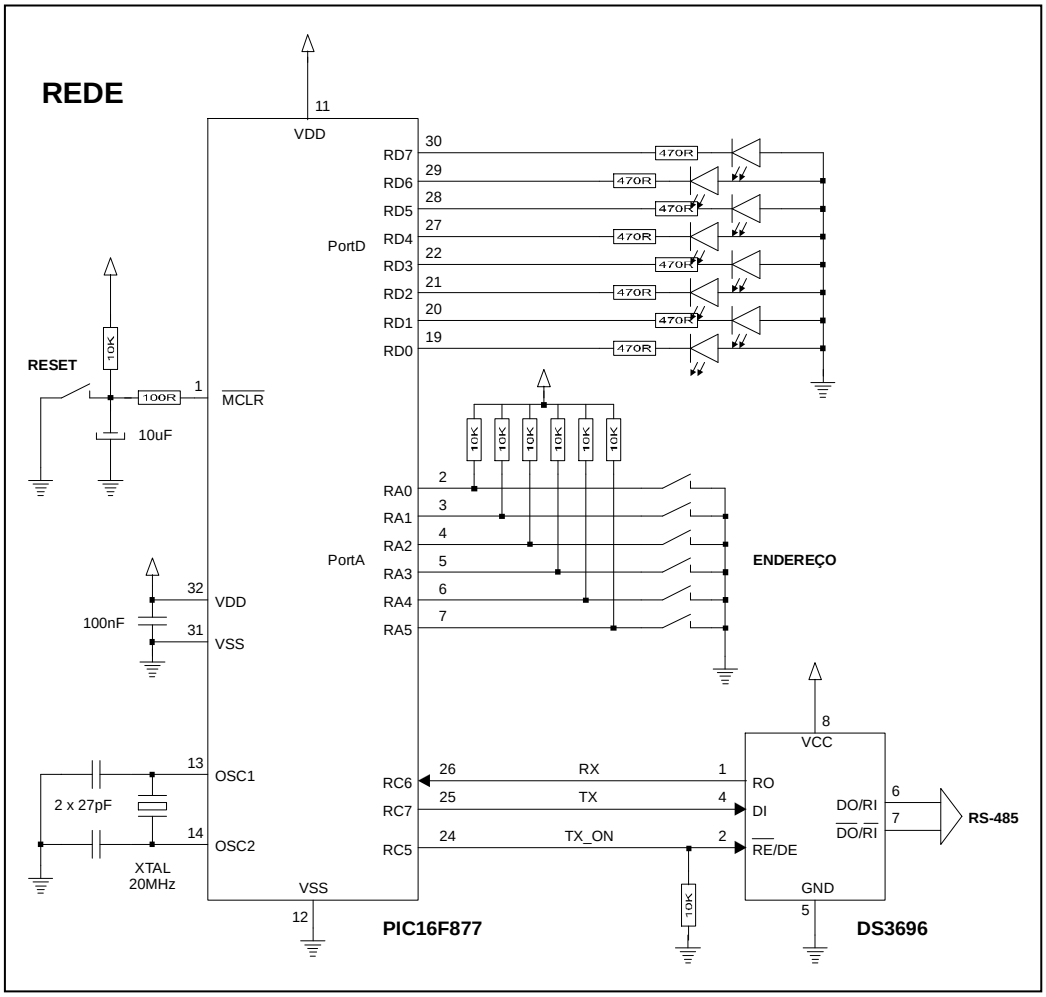
Primando pela velocidade, apenas testes internos do microcontrolador são realizados a fim de monitorar o recebimento de cada byte. Não foi considerado, portanto, o tratamento de erro de recepção, já que bytes errados representariam, teoricamente, apenas imagem distorcida.

4.3.1.3 HARDWARE

Uma vez determinadas as variáveis envolvidas no processo de transmissão, deve-se verificar o hardware necessário à transmissão. Isto porque é preciso definir as entradas e saídas que determinarão o funcionamento da interface.

O circuito integrado (CI) que servirá para converter o sinal digital do microcontrolador em padrão RS-485 foi escolhido pela disponibilidade e confiabilidade, além da existência da documentação necessária. O DS3696, conforme citado no capítulo 2.2.2 (circuito integrado para interface RS-485), necessita de um sinal de controle no pino 2. Um nível lógico alto (1) habilita o CI para transmitir, enquanto o nível baixo (0 – zero) deixa o componente em estado de recepção. Com base nessas informações, o circuito para implementação da rede pode ser visualizado na fig. 23.

FIGURA 23 – DIAGRAMA ELÉTRICO DO CIRCUITO PARA TESTE DA REDE

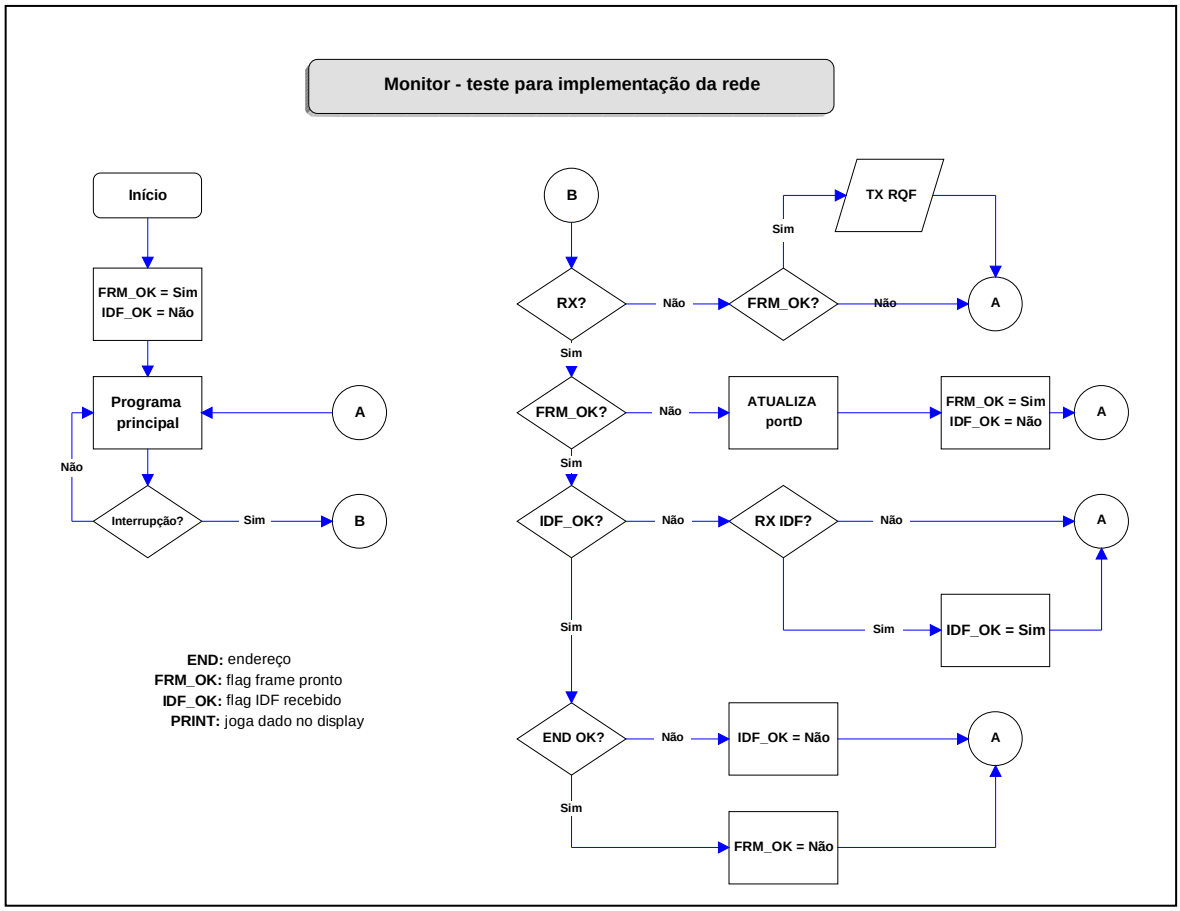


O portA do microcontrolador é responsável pela determinação do endereço, tanto de Monitor quanto de Câmera. Outro fator importante é a implementação de *led´s* no portD para facilitar a etapa de teste e validação.

4.3.1.4 ALGORITMO

Embora o circuito de teste de rede possa ser o mesmo, o algoritmo de Monitor e de Câmera são bem diferentes. O fluxograma da fig. 24 apresenta o algoritmo para desenvolvimento do software que ficará residente em Monitor, enquanto a fig. 25 mostra o algoritmo para a Câmera.

FIGURA 24 – MONITOR: ALGORITMO DA REDE

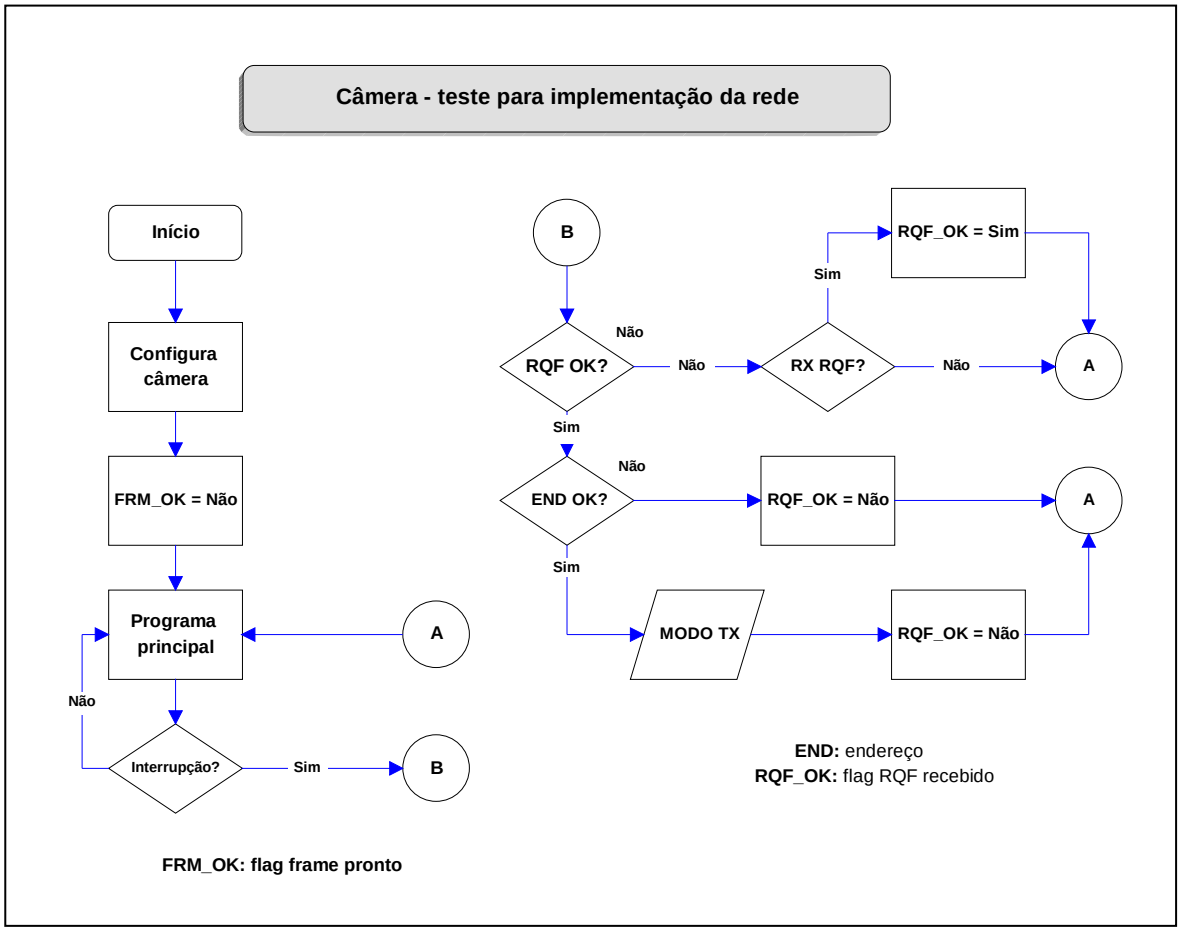


Ao receber a resposta ao comando RQF e confirmar o endereço de retorno, o Monitor passa a executar a subrotina “ATUALIZA portD” a cada byte recebido. Esta rotina compara os bytes recebidos ao endereço constante no portA, enquanto incrementa o portD e um contador que controla o tamanho do *frame*. Assim, quando Câmera envia o *frame*, o protocolo

de Monitor separa os dois primeiros bytes (comando e endereço) e conta os restantes pelo incremento do portD do Monitor. O algoritmo do contador, já em forma de código, será visto adiante.

A fig. 25 apresenta, então, o algoritmo para desenvolvimento do software que ficará residente em Câmera.

FIGURA 25 – CÂMERA: ALGORITMO DA REDE



A função MODO_TX transmite o comando IDF, o endereço em portA e 75Kbytes contendo o mesmo valor, lido no portA de Câmera. Os algoritmos apresentados até agora são praticamente os definitivos, dando assim já uma visão geral da implementação do software..

4.3.2 IMPLEMENTAÇÃO

Para a implementação da rede, como já citado no capítulo sobre ferramentas utilizadas, foram utilizados o MPLab e o Galep-III no desenvolvimento do software, além do *protoboard*, para montagem do hardware.

Para não tornar o trabalho demasiadamente extenso, serão apresentadas aqui apenas duas subrotinas que representam uma parte do código de cada implementação. Estas funções são pouco extensas e não sofrem posteriores alterações.

Como exemplo da implementação da rede em Monitor, vemos no quadro 4 a função TX_RQF.

QUADRO 4 – SUBROTINA TX_RQF

```
;
TX_RQF:                             ;tx requisicao de frame
        BCF         RCSTA,CREN      ; desabilita recepcao
        BSF         TX_ON           ; habilita chip serial 485 a tx
                                    ; tx comando:
        MOVLW       RQF             ;  w recebe comando a ser tx
        MOVWF       DADO            ;  salva dado a ser tx
        CALL        TX              ;  chama rotina p/ tx
                                    ; tx endereco:
        MOVFW       PORTA           ;  w recebe endereco a ser tx
        MOVWF       DADO            ;  salva dado a ser tx
        CALL        TX              ;  chama rotina p/ tx
        BSF         RCSTA,CREN      ; habilita recepcao
        BCF         TX_ON           ; habilita rx chip interface serial
        RETURN                      ; encerra rotina
;
TX:                                 ;tx dado
        BSF         STATUS,RP0      ; -> seleciona banco1
        BTFSS       TXSTA,TRMT      ; testa se buffer estah vazio
        GOTO        TX              ; loop enquanto buffer cheio
        BCF         STATUS,RP0      ; -> seleciona banco0
        MOVF        DADO,W          ; w recebe dado a ser tx
        MOVWF       TXREG           ; tx dado
DELAY_TX:                           ; aguarda buffer vazio p/ prosseguir
        BSF         STATUS,RP0      ; -> seleciona banco1
        BTFSS       TXSTA,TRMT      ;  testa se buffer estah vazio
        GOTO        DELAY_TX        ;  loop enquanto buffer cheio
        BCF         STATUS,RP0      ; -> seleciona banco0
        RETURN                      ; encerra rotina
;
```

Outro exemplo da implementação é a subrotina CONTA, que figura tanto em Monitor quanto em Câmera e pode ser vista no quadro 5. O código final, assim como todos os fluxogramas da implementação tanto de Câmera quanto de Monitor são apresentado nos apêndices II e III.

QUADRO 5 – SUBROTINA CONTA

```
;
CONTA:                                  ;contador (limites definidos por constantes)
        INCFSZ      CONT0,F             ; incrementa lsb e testa: igual a 0?
        GOTO        $+2                 ;  nao, salta p/ testar cont1 e cont0
        GOTO        INC_C1              ;  sim, incrementa cont1
        BTFSS       C1_OK               ; cont1 = lim_c1?
        RETURN                          ;  nao, encerra rotina.
        MOVF        LIM_C0,W            ;  sim, w recebe limite de cont0
        XORWF       CONT0,W             ; compara contador 0 c/ valor limite
        BTFSS       STATUS,Z            ; cont0 = lim_c0?
        RETURN                          ;  nao, encerra rotina.
                                        ; sim, cont1 = lim_c1 e cont0 = lim_c0
        BSF         FDT                 ; seta flag fim de tela
        BCF         C1_OK               ; zera flag (cont1 = lim_c1)
        CLRF        CONT0               ; limpa lsb contador
        CLRF        CONT1               ; limpa msb contador
        RETURN                          ;encerra rotina.
INC_C1                                  ; incrementa cont1
        INCF        CONT1,F             ; incrementa byte msb do contador
        MOVF        LIM_C1,W            ;  sim, w recebe limite de cont1
        XORWF       CONT1,W             ; compara contador 1 c/ valor limite
        BTFSS       STATUS,Z            ;  cont1 = lim_c1?
        RETURN                          ;   nao, entao encerra rotina.
        BSF         C1_OK               ;   sim, seta flag (cont1 = lim_c1)
        RETURN                          ; encerra rotina.
;
```

4.3.3 TESTES E VALIDAÇÃO

Os testes foram realizados com e sem a implementação do CI de interface (DS3696), com os mesmos resultados alcançados.. Foram alcançadas taxas da ordem de 1,25Mbps, permitindo a transferência do conteúdo de um *frame* hipotético (formado por valores fixos na entrada – portD) a cada 700ms. A validação dos dados recebidos foi comprovada pela visualização do padrão de acionamento dos *led´s* no portD do hardware Monitor. Enquanto o endereço no portA de Monitor corresponda ao endereço no portA de uma Câmera ativa, o portD em Monitor será atualizado.

A verificação dos tempos foi feita com auxílio de osciloscópio, medindo-se o intervalo entre cada nova transmissão do comando RQF (no caso do Monitor), ou então, calculando-se o tempo gasto para transmitir um byte e multiplicando-se pela quantidade de bytes do *frame* (no caso da Câmera).

E, embora não tenha sido implementada detecção de colisão, testes foram feitos utilizando-se dois Monitores e uma Câmera, alcançando-se resultados positivos. Face ao que foi proposto e constatado, a implementação da rede foi dada como satisfatória.

4.4 MONITOR

Estando a rede já em operação, o *display* pode ser acrescentado em Monitor. Para isso, é necessário alteração de hardware e a correta configuração do módulo LCD, tendo o cuidado de analisar os tempos envolvidos nos ciclos de escrita e leitura.

4.4.1 ESPECIFICAÇÃO

O software do Monitor deve sofrer uma pequena alteração na parte da rede. Não são mais 75Kbytes por *frame*, já que o *display* adotado no protótipo possui apenas 160 colunas por 80 linhas. Além disso, ele é monocromático, ou seja, cada pixel é representado por apenas 1 bit. Então, uma matriz de 160 por 80 totaliza 12800 pontos que, divididos por 8 bits faz um total de 1600 bytes apenas.

Entretanto, para adiantar um pouco a especificação, deve ser considerado também o formato do *frame* transmitido pela câmera digital. No capítulo 2.3.1 (módulo de câmera digital), mostra que a janela de captura de imagem é dividida em pequenos blocos de 18 pixels de altura por 24 pixels de comprimento. O maior formato, múltiplo destes tamanhos de bloco, que caberia diretamente no *display* adotado, seria 144 colunas por 72 linhas. Isto faz com que cada linha do *display* tenha 18 bytes, num total de 10368 pixels ou 1296 bytes a serem recebidos de Câmera.

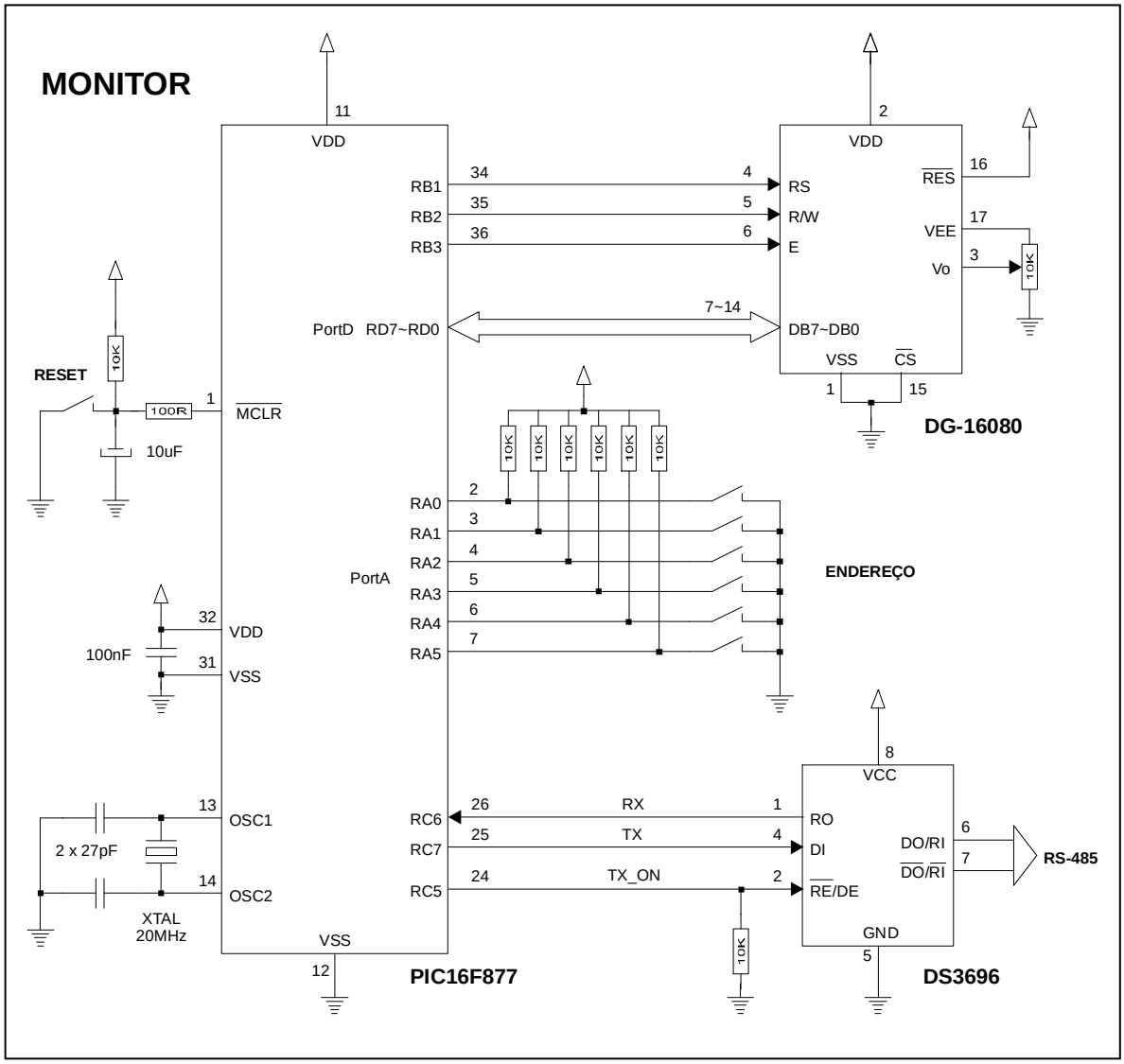
4.4.1.1 HARDWARE

É necessário também implementar os sinais de controle do *display* e substituir os *led´s* no portD pelo barramento bidirecional de dados (fig. 26). Assim, ao invés de acender *led´s*, o

Monitor passa a mostrar barras verticais no *display*. Resultado ainda da configuração das chaves no portD da Câmera, transmitidos ao Monitor já via RS-485.

Os sinais de controle implementados são descritos no capítulo 2.4.3 (*display* gráfico).

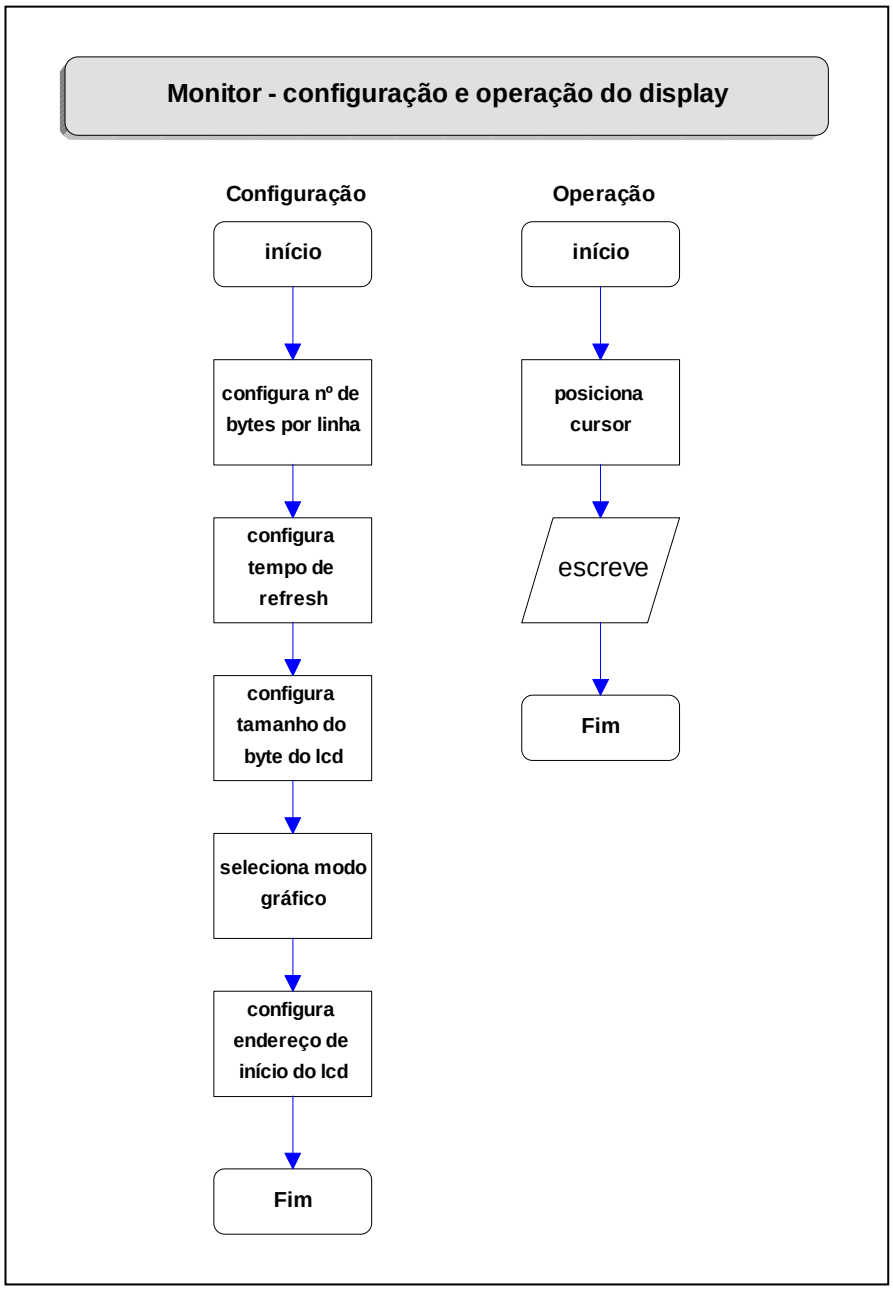
FIGURA 26 – ESQUEMA ELÉTRICO DO MONITOR



4.4.1.2 CONFIGURAÇÃO DO *DISPLAY*

A configuração do *display* é feita na inicialização do hardware Monitor, e procede conforme o fluxograma da fig. 27.

FIGURA 27 – FLUXOGRAMA CONFIGURAÇÃO E OPERAÇÃO DO *DISPLAY*



4.4.2 IMPLEMENTAÇÃO

Aqui, demonstrando a etapa de implementação, pode ser visto no quadro 6 a parte do arquivo fonte relativo à configuração do *display*, onde os parâmetros são passados na ordem sugerida pelo manual do fabricante e determinados pelo cálculo do tamanho do *frame*.

QUADRO 6 – ROTINA DE CONFIGURAÇÃO DO *DISPLAY*

```
                                ;configura display:
                                ; qtde bytes por linha
MOVLW       H'02'               ;  w    <- 02h (reg. numero caracteres)
MOVWF       INST                ;  INST <- w
MOVLW       D'17'               ;  w    <- 17 (18 bytes/linha)
MOVWF       OPER                ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ; divisao de tempo - refresh
MOVLW       H'03'               ;  w    <- 03h (registro divisoes tempo)
MOVWF       INST                ;  INST <- w
MOVLW       D'79'               ;  w    <- 79
MOVWF       OPER                ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ; qtde bits por palavra (tamanho do byte)
MOVLW       H'01'               ;  w    <- 01h (registro tamanho caracter)
MOVWF       INST                ;  INST <- w
MOVLW       H'77'               ;  w    <- 77h (07h - 8 bits por byte)
MOVWF       OPER                ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ; modo gráfico
MOVLW       H'00'               ;  w    <- 00h (registro controle de modo)
MOVWF       INST                ;  INST <- w
MOVLW       H'32'               ;  w    <- 32h (modo grafico - master)
MOVWF       OPER                ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ; endereco inicial display (low)
MOVLW       H'08'               ;  w    <- 08h (reg. endereco inicial LOW)
MOVWF       INST                ;  INST <- w
MOVLW       H'00'               ;  w    <- 00h
MOVWF       OPER                ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ; endereco inicial display (high)
MOVLW       H'09'               ;  w    <- 09h (reg. endereco inicial HIGH)
MOVWF           INST            ;  INST <- w
MOVLW           H'00'           ;  w    <- 00h
MOVWF           OPER            ;  OPER <- w
CALL        EXELCD              ;  executa operacao no lcd
                                ;
```

No quadro 7 estão presentes as subrotinas usadas para efetivar a configuração. Detalhe especial para a subrotina LE_FLAG, que faz uma verificação em um flag do módulo antes de proceder cada uma das operações necessárias à inicialização ou operações de escrita do *display*.

QUADRO 7 – SUBROTINAS DE CONFIGURAÇÃO DO *DISPLAY*

```
;
EXELCD:                                 ;joga instrucao/operando no lcd p/ execucao
                                        ;escreve instrucao
        CALL        LE_FLAG             ;testa lcd busy
        MOVF        INST,W              ; W <- INST
        MOVWF       PORTD               ; escreve codigo da instrucao
        BCF         RW                  ; zera RW - escrita
        BSF         RS                  ; seta RS - instrucao
        BSF         E                   ; seta E - habilita operacao escrita
        NOP                             ; delay p/ lcd assimilar comando
        BCF         E                   ; zera E - aguarda proxima operacao
                                        ;escreve operando
        CALL        LE_FLAG             ; testa lcd busy
        MOVF        OPER,W              ; W <- OPER
        MOVWF       PORTD               ; escreve operando referente a instrucao
        BCF         RW                  ; zera RW - escrita
        BCF         RS                  ; zera RS - operando
        BSF         E                   ; seta E - habilita operacao escrita
        NOP                             ; delay p/ lcd assimilar comando
        BCF         E                   ; zera E - aguarda proxima operacao
        RETURN                          ;encerra rotina
;
LE_FLAG:                                ;le flag lcd busy
        BSF         STATUS,RP0          ; -> seleciona banco1
        BSF         TRISD,7             ; configura bit7 do portD como entrada
        BCF         STATUS,RP0          ; -> seleciona banco0
BUSY:
        BCF         E                   ; zera E - aguarda proxima operacao
        BSF                 RS          ; seta RS - instrucao
        BSF                 RW          ; seta RW - leitura
        BSF                 E           ; seta E - habilita operacao leitura
        BTFSC       BSY                 ; testa flag lcd busy - lcd ready?
        GOTO        BUSY                ;  nao, loop enquanto lcd busy
        BCF                 E           ; zera E - aguarda proxima operacao
        BSF                 STATUS,RP0  ; -> seleciona banco1
        BCF                 TRISD,7     ; configura bit7 do portD como saida
        BCF                 STATUS,RP0  ; -> seleciona banco0
        RETURN                          ; encerra rotina
;
```

4.4.3 TESTES E VALIDAÇÃO

Os testes tiveram resultado positivo. O padrão de faixas verticais visualizados no *display* é rapidamente alterado, tão logo é feita modificação nas chaves do portD do hardware ainda experimental de qualquer Câmera endereçada. Este será o circuito definitivo de Monitor, já que com a rede e o *display* implementados, e em funcionamento, pode-se considerar a etapa Monitor concluída.

4.5 CÂMERA

Para a anexação da câmera ao protótipo, são necessárias algumas alterações a nível de hardware e de software. Feito isto, o protótipo pode finalmente ser implementado e testado a nível de conjunto final.

4.5.1 ESPECIFICAÇÃO

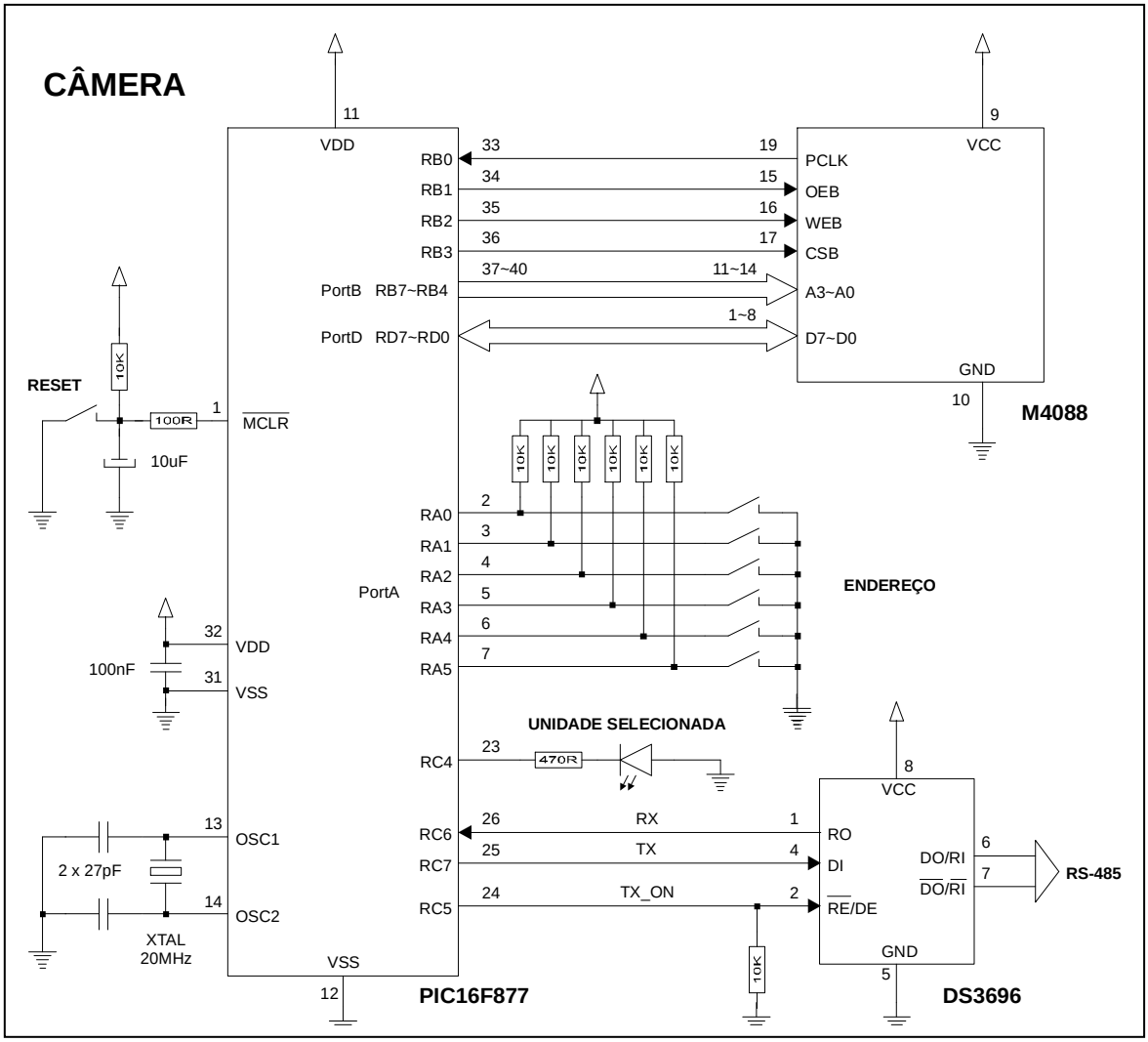
A última parte da implementação do protótipo exige mais sinais de controle por parte do microcontrolador. Uma solução a nível de software também é proposta, a fim de resolver um problema gerado por diferente representação do pixel por parte de câmera e *display*

4.5.1.1 HARDWARE

Os sinais de controle necessários ao correto funcionamento da câmera digital são descritos no capítulo 2.3.1 (módulo de câmera digital M4088).

As alterações no circuito elétrico, na verdade, são apenas acréscimos ao hardware definido para Monitor. Desta forma, tanto Monitor quanto Câmera poderiam ser construídos a partir do mesmo *layout* de placa de circuito impresso, o que representaria economia a nível de implementação. A fig. 28 demonstra essa tese.

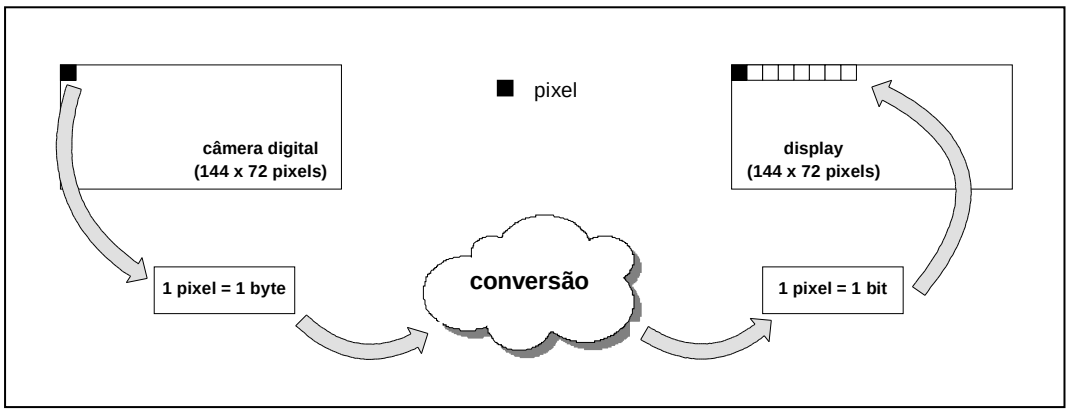
FIGURA 28 – ESQUEMA ELÉTRICO DA CÂMERA



4.5.1.2 REPRESENTAÇÃO DO PIXEL

Já que a programação da câmera obedece a padrão semelhante à programação do *display*, a nível de implementação, a principal alteração é uma adaptação para adequar a representação da imagem por parte do Monitor em relação à captura por parte da Câmera. A fig. 29 mostra esquematicamente o problema encontrado.

FIGURA 29 – CAPTURA E REPRESENTAÇÃO DA IMAGEM



Para resolver o problema, cada pixel lido na câmera digital deve ser convertido em um bit, preferencialmente antes de ser transmitido para Monitor. Como os valores fornecidos pela câmera digital ficam entre 00h a FFh, inclusive, o meio termo dessa faixa seria 80h (10000000b). Logo, basta testar o bit mais significativo do byte lido da câmera digital para definir o valor do bit convertido. Assim:

a) se o bit de ordem sete do byte lido for 1 (um), o pixel no *display* deve ser “ligado”;
b) se o bit de ordem sete do byte for 0 (zero), o pixel no *display* será “desligado”.

O código referente ao algoritmo proposto é mostrado no item “implementação” e o fluxograma é mostrado no apêndice II, juntamente com os demais fluxogramas do protótipo final.

4.5.2 IMPLEMENTAÇÃO

Em acréscimo ao que já foi implementado a nível de algoritmo, a subrotina MONTA_BYTE é mostrada no quadro 8. Ela faz a conversão dos pixels lidos pela câmera digital em conjuntos de oito pixels para serem transmitidos ao Monitor.

QUADRO 8 – SUBROTINA DE CONVERSÃO DE PIXEL

```
;
MONTA_BYTE                              ;transforma o byte lido em um bit
            CLRF        CONT            ; inicializacao do contador de bits
            CLRF        DADO            ; inicializacao da variavel aux
CONV_BIT                                ;repete p/ cada bit do byte lido
            MOVF        PORTB,W         ; le valor pixel
            ANDRLW      B'1000000'      ; compara w com B'1000000'
            BCF         STATUS,C        ; zera carry p/ rotacionar
            BTFSS       STATUS,Z        ; bit7 ligado?
            BSF         STATUS,C        ;  sim, seta carry p/ rotacionar
            RLF         DADO,F          ; rotaciona bit de dado p/ esquerda
            INCF        CONT,F          ; incrementa contador
            BTFSC       CONT,3          ; byte montado? (testa bit ordem 3)
            RETURN                      ;  sim, encerra      rotina
            GOTO        CONV_BIT        ;  nao, converte proximo bit
;
```

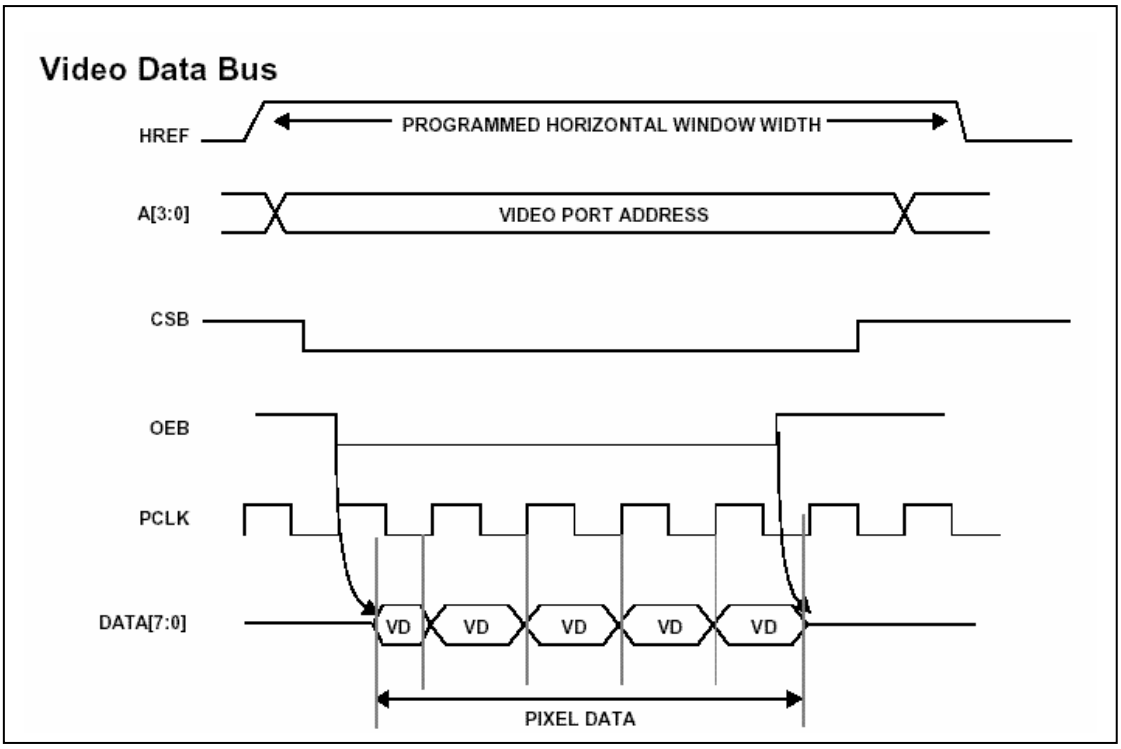
4.5.3 TESTES E VALIDAÇÃO

A programação da câmera procedeu da maneira prevista, indicada por testes individuais com o osciloscópio. Porém, nenhuma imagem pode ser visualizada durante os testes para validação da implementação do hardware Câmera.

Apesar das alterações do software na tentativa de capturar imagens, os estes realizados indicaram problemas relativos a sincronismo na captura, gerando erros de transmissão e imagens disformes.

Medições feitas com o auxílio do osciloscópio mostram que os tempos dos pulsos de sincronismo de pixel (PCLK) giram em torno de 9µs (fig. 30), contra 6,8µs de cada pixel transmitido. Os 2,2µs restantes não são suficientes para realizar a conversão necessária do pixel, acarretando perda de dados e gerando os problemas de sincronismo de leitura.

FIGURA 30 – DIAGRAMA DE TEMPOS PARA AQUISIÇÃO DA IMAGEM



Fonte: Omnivision Technologies (1998).

Alterações feitas no sentido de enviar os bytes diretamente ao Monitor, sem conversão, surtiram efeito. Nesse caso, o que é visto na tela são os pixels enviados, em forma de pontos, não sendo possível a visualização da imagem.

4.6 RESULTADOS E DISCUSSÃO

As etapas Rede e Monitor mostraram-se satisfatórias, sendo comprovados os resultados obtidos através de instrumentação (osciloscópio) e simulações com valores fixos transmitidos a partir do hardware Câmera.

Com a implementação do *watchdog* no código, a rede tornou-se ainda mais estável, pois mesmo com a desconexão de qualquer dos módulos durante a transmissão, a mesma continua operando normalmente.

Outro fato importante é a rápida atualização da imagem quando solicitada à uma Câmera em operação. Em caso de endereçamento incorreto, ou Câmera inoperante, o *display* é limpo, como sinal da ausência de recepção.

Devido a problemas com o sincronismo de dados entre microcontrolador e câmera digital, no hardware Câmera, não foi possível determinar se a qualidade das imagens apresentadas seriam aceitáveis.

Uma maneira de contornar o problema seria o acréscimo de uma expansão de memória externa, que fosse capaz de armazenar um *frame* inteiro. A operação de Câmera seria então alterada para capturar um *frame* nos intervalos de transmissão. Dessa forma o ritmo de captura e de transmissão poderiam ser desacoplados, permitindo que a imagem fosse corretamente visualizada em Monitor.

Outras formas de contornar o problema seriam:

a) usar um microcontrolador capaz de executar as instruções necessárias à conversão do pixel em tempo real;
b) implementar no Monitor um *display* que tenha o mesmo tipo de representação do *pixel* que o usado pela câmera digital, isto é, um *pixel* igual a um byte.

5 CONCLUSÕES

Foi constatado um fato citado na bibliografia vista no início das pesquisas: o surgimento de um outro ramo de atividade ligada principalmente à automação residencial. O integrador de sistemas, que é um profissional com conhecimentos de computação e eletrônica, e cuja missão principal é adaptar sistemas prontos, criando soluções que vão de encontro às necessidades do usuário.

Este trabalho demonstrou exatamente a interligação entre três sistemas distintos: um microcontrolador complexo e versátil, um módulo de câmera digital programável e um módulo LCD igualmente configurável. Cada um com recursos próprios, sendo imprescindível compatibililizar as informações necessárias a sua interligação.

As ferramentas utilizadas foram de vital importância ao desenvolvimento do trabalho, citando-se principalmente o *protoboard*, o osciloscópio e o ambiente de desenvolvimento MPLab – apesar deste último não simular algumas funções importantes como as da USART. A tarefa mais árdua foi, sem dúvida, a aquisição do conhecimento necessário à implementação do protótipo. Exemplo disso é a pouca bibliografia, existente no ciclo acadêmico, falando especificamente sobre visualização de imagem em *display*s de cristal líquido. Esse conteúdo foi encontrado apenas em páginas da Internet. Isto ocorreu da mesma forma com os módulos de câmera digital.

A pouca informação inicial, remediada com experiências práticas, obrigaram por exemplo, a substituição de um *display* de 240 por 200 *pixels*, que tornaria a imagem um pouco maior, por um de 160 por 80 *pixels* – tamanho disponível no momento da implementação. Isto só aconteceu depois de muitas tentativas de implementação utilizando-se o microcontrolador adotado no protótipo. Mas a facilidade de se manipular um *display* com controlador incorporado é tamanha que não vale o esforço, se não houver a necessidade.

Outro problema acarretado pela pouca informação inicial foi o detectado durante a implementação da etapa Câmera. A incompatibilidade de representação do pixel entre display e câmera digital acabaram por inviabilizar o teste de visualização de imagem, já que as alterações necessárias despenderiam mais tempo para implementação, tempo inexistente devido às circunstâncias.

É necessário acrescentar que foram realizados estudos para implementação do modo de transmissão síncrono, que permitiram taxas de transmissão de 5Mbps – comprovados em laboratório. Porém, o modo síncrono não é suportado pelo padrão RS-485.

Entretanto, os objetivos podem ser considerados válidos, no momento em que se analisa os resultados alcançados. Taxas de transmissão de imagem da ordem de 1,25 Mbps com frames sendo transmitidos a cada 700ms, a utilização correta do módulo LCD em modo gráfico, além da implementação da rede de comunicação de dados e o aprofundamento da codificação em *assembly*. Todas essas etapas foram realizadas com sucesso.

5.1 EXTENSÕES

O trabalho foi proposto a um fim específico: um sistema de vigilância através de monitoração constante de pontos pré definidos em um ambiente. Mas também poderia ser encarado como um subsistema de segurança, interagindo com uma central que gerenciaria outras funções na residência. O controle de iluminação, sonorização, climatização, assim como gerenciamento de energia, telecomunicações e água, reforçam a idéia da total automação do ambiente - conceito esse defendido na totalidade da bibliografia específica pesquisada.

Dessa forma, além das correções citadas no capítulo 4.6 (resultados e discussão), uma sugestão seria a implementação de comunicação com microcomputadores, a fim de possibilitar a interação necessária ao desenvolvimento de um sistema mais abrangente de automação residencial.

Além disso, as taxas de transmissão ainda poderiam ser melhoradas utilizando-se de microcontroladores mais rápidos e/ou tecnologias como USB e *firewire* para a transmissão dos dados.

GLOSSÁRIO

BACKLIGHT – iluminação utilizada para facilitar a visualização de módulos LCD em ambientes com pouca luz.

CAPACITÂNCIA – capacidade que um determinado corpo tem de acumular cargas elétricas.

CAPACITÂNCIA PARASITA – é o acúmulo indesejado, ou não quantificado, de cargas elétricas em um determinado corpo.

CAPACITOR – componente eletrônico especificamente dotado de capacidade de armazenar cargas elétricas.

CHECKSUM – código gerado por algoritmo especializado para identificar a fim de comparar blocos de dados.

CHIP – pastilha de silício contendo milhares de componentes que compõe os CI's.

DATASHEET – especificação técnica fornecida pelo fabricante de determinado componente.

EMULAÇÃO – uso de um equipamento no lugar de um dispositivo (microcontrolador, memória) exercendo as mesmas funções deste, no intuito de facilitar o desenvolvimento/depuração do software, antes da gravação da versão final.

FIREWIRE – especificação de barramento serial para comunicação de dados (IEEE 1394), semelhante ao USB, criado pela empresa *Apple* na década de 90.

FLASH – memória não volátil semelhante a EEPROM que difere desta por não haver necessidade de regravação total quando de uma alteração de dados.

FRAME – quadro; imagem parada, sem movimento; é usado também em comunicação de dados para especificar o formato da informação transmitida.

IN-CIRCUIT – termo usado para designar a gravação/emulação de dispositivo diretamente na placa, sem a necessidade de retirá-lo do lugar.

INTERFACE DIGITAL SERIAL DIFERENCIAL BALANCEADA – especificação de sinais elétricos para comunicação de dados.

LCD – visor, monitor ou *display* de cristal líquido.

LED – diodo emissor de luz; normalmente nas cores verde, vermelho e amarelo, é usado para sinalização luminosa em diversos equipamentos eletrônicos.

PINAGEM – nome dos pinos de um determinado circuito ou dispositivo.

PIXEL – do inglês *picture elements*; elemento de imagem; ponto, menor elemento que compõe a imagem.

PLACA DE CAPTURA DE VÍDEO – cartões eletrônicos instalados em sistemas computacionais no intuito de transformar o sinal analógico proveniente de câmeras de vídeo em sinal digital, para ser processado pelo computador.

PRESCALER – divisor usado para aumentar escala de tempo em temporizadores.

REFRESH – reescrita periódica da imagem em módulos LCD, para mantê-la visível; técnica também usada em alguns tipos de memória para manter dados armazenados.

RESISTOR – componente eletrônico capaz de fornecer resistência à passagem de corrente elétrica, medido em ohms.

USB – *Universal Serial Bus*, especificação de um padrão de barramento serial universal para comunicação de dados.

WATCHDOG – cão de guarda; temporizador responsável por reinicializar o microcontrolador em caso de travamento do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

AS CASAS do século XXI. **Exponornews**, São Paulo, n. 8, jul. 2001. Publicação periódica da Exponor Brasil Ltda. p. 7.

AURESIDE. **Peculiaridades dos sistemas de automação residencial**, São Paulo, set. 2001. Disponível em: <http://www.aureside.org.br>. Acesso em: 28 set. 2001.

ATOS. **Boletim técnico EP-03/00:** informações básicas para implantação de uma rede de controladores, São Paulo, fev. 2000. Disponível em: <http://www.atos.com.br/download/default.asp>. Acesso em: 23 nov. 2001.

COMEDIA. **d-m4088.pdf.** M4088 1/4" B/W camera module with digital output, [S.l.], 09 nov. 1999. Arquivo baixado. 90Kb. Acrobat Reader. Disponível em: <http://www.electronics123.com/amazon/datasheet/d-m4088.pdf>. Acesso em: 07 mai. 2002.

CORRIGAN, John. **Computação gráfica:** segredos e soluções. Rio de Janeiro: Ed. Ciência Moderna, 1994.

CUNHA, Judson Michel. **Protótipo de rede industrial utilizando o padrão serial RS485 e protocolo MODBUS.** 2000. 133 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

DATA INTERNACIONAL. **Mono STN graphic**, [S.l.], jun. 2002. Disponível em: <http://www.datainternational.com/pro_monoStn.php3>. Acesso em: 15 jun. 2002.

DOCTRONICS. **1P:** prototype circuits, [S.l.], [2002?]. Disponível em: <http://www.doctronics.co.uk/prototyp.htm>. Acesso em: 24 mai. 2002.

EXCEL TECHNOLOGY. **16080.pdf.** DG-16080 data sheet, [S.l.], 15 jun. 2002. Arquivo baixado. 55Kb. Acrobat Reader. Disponível em: <http://www.lcdexceltech.com/lcmpdf/datasheet/16080.pdf>. Acesso em: 15 jun. 2002.

EXCEL TECHNOLOGY. **Excelix graphic LCM**, [S.l.], [S.d..]. Disponível em: <http://www.datainternational.com/pro_monoStn.php3>. Acesso em: 15 jun. 2002.

FOLEY, James D. **Computer graphics:** principles and practice. 2.ed. Reading: Addison-Wesley, 1990.

FRANCO, Lúcia Regina Horta R. **EIA RS-485 field bus**, Itajubá, [2001?]. Disponível em: http://www.iee.efei.br/~gaii/rs485/hp_rs485.htm. Acesso em: 23 nov. 2001.

HANTRONIX. **3224app.pdf.** Interfacing a Hantronix 320 x 240 graphics module to an 8-bit microcontroller, Cupertino, 14 fev. 2000. Arquivo baixado. 59Kb. Acrobat Reader. Disponível em: <http://www.hantronix.com/down/3224app.pdf>. Acesso em: 27 mai. 2002.

HANTRONIX. **S240200.pdf.** Product specification HDM240200 240 x 200 graphics LCD display module, Cupertino, 21 set. 1998. Arquivo baixado. 182Kb. Acrobat Reader. Disponível em: <http://www.hantronix.com/down/s240200.pdf>. Acesso em: 27 mai. 2002.

INFORMÁTICA MÉDICA. **Processamento de imagens pelo computador**, Campinas, dez. 1998. Disponível em: <http://www.epub.org.br/informaticamedica/n0106/imagens.htm.> Acesso em: 21 mai. 2002.

INTEGRITY INSTRUMENTS. **Technical support - data communication specifications,** Pine River, [2001?]. Disponível em: <http://www.integrityusa.com/specs.htm>. Acesso em: 03 dez. 2001.

INTEL. **MCS-51:** family of single chip microcomputers user´s manual. Santa Clara: Intel, 1981.

KLITZKE, Marcelo. **Protótipo de hardware para aquisição e transmissão de imagens via padrão serial RS-485**. 1999. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MARTE, Cláudio Luiz. **Automação predial:** a inteligência distribuída nas edificações. São Paulo: Carthago & Forte, 1995.

MICROCHIP. **30292c.pdf.** PIC16F87X data sheet - 28/40-pin 8-bit CMOS FLASH microcontrollers, Chandler, 05 fev. 2001. Arquivo baixado. 3,92Mb. Acrobat Reader. Disponível em: <http://www.microchip.com/download/lit/pline/picmicro/families/16f87x/30292c.pdf>. Acesso em: 16 jun. 2002.

MICROCHIP Technology. **AN520.pdf.** AN520 – A comparison of 8-bit microcontrollers, Chandler, 24 jun. 1996. Arquivo baixado. 71Kb. Acrobat Reader. Disponível em: <http://www.eetasia.com/ART_8800046991_499481,499490.HTM>. Acesso em: 16 jun. 2002.

MICROCHIP Technology. **mpl5000.zip.** MPLAB 5000 version 5.00.00, Chandler, 09 mar. 2000. Arquivo baixado. 8,96Mb. Winzip. Disponível em: <http://www.microchip.com/download/tools/archive/mplabi/mpl5000.zip>. Acesso em: 16 jun. 2002.

MINIPA. **Produtos**, São Paulo, mai. 2002. Disponível em: <http://www.minipa.com.br>. Acesso em: 24 mai. 2002.

NAGANO, Mário. **Fotografia digital sai da infância e alcança a adolescência**, [S.l.], fev. 2000. Disponível em: <http://pcworld.terra.com.br/pcw/testes/cameras_digitais/0002.html>. Acesso em: 21 mai. 2002.

NATIONAL SEMICONDUCTOR. **DS3695.pdf.** DS3695/DS3695T/DS3696/DS3697 Multipoint RS485/RS422 Transceivers/Repeaters, [S.l.], 06 out. 1998. Arquivo baixado. 310Kb. Acrobat Reader. Disponível em: <http://www.national.com/pf/DS/DS3696.html>. Acesso em: 23 nov. 2002.

NUNES, José Renato Soares. **Comunicação de dados:** conceitos básicos. Rio de Janeiro: LTC, 1989.

OKI SEMICONDUCTOR. **Faq´s – lcd drivers**, [S.l.], [2002?]. Disponível em: <http://www.okisemi.com/public/docs/LCDFaq.html>. Acesso em: 31 mai. 2002.

OMNIVISION TECHNOLOGIES. **OV5017DS.pdf.** OV5017, Sunnyvale, 03 set.. 1998. Arquivo baixado. 195Kb. Acrobat Reader. Disponível em:

<http://www.electronics123.com/amazon/datasheet/OV5017DS.pdf>. Acesso em: 07 mai. 2002.

SANYO. **LC7981.pdf.** LCD dot matrix graphic display controller, [S.l.], [S.d.]. Arquivo baixado. 387Kb. Acrobat Reader. Disponível em: <http://www.hantronix.com/down/lc7981.pdf>. Acesso em: 27 mai. 2002.

SEIKO Epson corporation. **SEC1335.pdf.** SED 1335 series lcd controller ics: technical manual, [S.l.], 12 mar. 1999. Arquivo baixado. 399Kb. Acrobat Reader. Disponível em: <http://www.elateceurope.com/displays/pdf/sed1335.pdf>. Acesso em: 31 mai. 2002.

SILVA JÚNIOR, Vidal Pereira da. **Microcontrolador 8051:** hardware e software. São Paulo: Livros Érica, 1990.

SILVA JÚNIOR, Vidal Pereira da. **Microcontroladores PIC:** teoria e prática. São Paulo: Vidal Pereira da Silva Júnior, 1997.

SOUZA, David José de. **Desbravando o PIC**. 2. ed. São Paulo: Érica, 2000.

SMITH, R. E. **Serial communications RS485**. Hamilton, mar. 1999. Disponível em: <http://www.rs485.com>. Acesso em: 03 dez. 2001.

TAFNER, Malcon Anderson; LOESCH, Claudio; STRINGARI, Sérgio. **Comunicação de dados usando linguagem C:** aplicação em DOS e Windows. Blumenau: Ed. Da FURB, 1996.

TECH DATA. **Monitor IBM TFT com alteração de tela de 90° chega ao Brasil**, [S.l.], [1999?]. Disponível em: <http://www.techdata.com.br/novidades/news143.htm>. Acesso em: 31 mai. 2002.

TEKTRONIX. **Osciloscópios**, [S.l.], [2002?]. Disponível em: <http://www.tektronics.com.br>. Acesso em: 24 mai. 2002.

TORRES, Gabriel. **Barramento firewire (IEEE 1394)**, [S.l.], jun. 2000. Disponível em: <http://www.clubedohardware.com.br/firewire.html>. Acesso em: 12 nov. 2001.

ANEXO I – PIC16F877: DESCRIÇÃO DOS PINOS

Mnemônico	N.º	Tipo	Descrição
OSC1/CLKIN	13	I (ST/CMOS) [4]	Entrada para cristal oscilador ou clock externo.
OSC2/ CLKOUT	14	O	Saída para cristal oscilador. Conectado ao cristal ou ressonador no modo cristal oscilador. No modo RC, fornece ¼ da freqüência de OSC1, indicando taxa de ciclo de instrução.
~~MCLR~~/VPP	1	I/P (ST)	Entrada Master Clear (Reset) ou entrada de voltagem para programação in-circuit. É ativado com nível baixo (0 Vcc).
VSS	12,31	P	Alimentação do chip (0Vcc).
VDD	11,32	P	Alimentação do chip (5Vcc).
A Porta A é uma porta de entrada/saída bidirecional.			
RA0/AN0	2	I/O (TTL)	RA0 também pode ser entrada analógica 0.
RA1/AN1	3	I/O (TTL)	RA1 também pode ser entrada analógica 1.
RA2/AN2/ Vref-	4	I/O (TTL)	RA2 também pode ser entrada analógica 2 ou voltagem de referência analógica negativa.
RA3/AN3/ Vref+	5	I/O (TTL)	RA3 também pode ser entrada analógica 3 ou voltagem de referência analógica positiva.
RA4/T0CKI	6	I/O (ST)	RA4 também pode ser entrada de clock para Timer0. Como saída, é do tipo dreno aberto.
RA5/~~SS~~/AN4	7	I/O (TTL)	RA5 também pode ser entrada analógica 4 ou seleção de escravo para porta serial síncrona. ~~SS~~ é ativado com nível baixo (0 Vcc).
A Porta B é uma porta de entrada/saída bidirecional, com weak pull-up habilitado via software para todas as entradas.			
RB0/INT	33	I/O (TTL/ST) [1]	RB0 também pode ser pino de interrupção externa.
RB1	34	I/O (TTL)	
RB2	35	I/O (TTL)	
RB3/PGM	36	I/O (TTL)	RB3 também pode ser entrada de voltagem de referência (0 Vcc) para programação in-circuit (vide ~~MCLR~~, pino 1).
RB4	37	I/O (ST)	Interrupção de mudança de estado.
RB5	38	I/O (TTL)	Interrupção de mudança de estado.
RB6/PGC	39	I/O (TTL/ST) [2]	Interrupção de mudança de estado ou pino para depuração in-circuit. Clock para programação serial.
RB7/PGD	40	I/O (TTL/ST) [2]	Interrupção de mudança de estado ou pino para depuração in-circuit. Dados para programação serial.
A Porta C é uma porta de entrada/saída bidirecional.			
RC0/T1OSO/ T1CKI	15	I/O (ST)	RC0 também pode ser saída do oscilador do Timer1 ou uma entrada de clock para Timer1.
RC1/T1OSI/ CCP2	16	I/O (ST)	RC1 também pode ser entrada do oscilador do Timer1 ou ainda entrada de captura 2, saída do comparador 2 ou saída PWM2.
RC2/CCP1	17	I/O (ST)	RC2 também pode ser entrada de captura 1, saída do comparador 1 ou saída PWM1.
RC3/SCK/ SCL	18	I/O (ST)	RC3 também pode ser entrada de clock serial síncrono ou saída para os modos SPI e I²C.
RC4/SDI/ DAS	23	I/O (ST)	RC4 também pode ser entrada de dados no modo SPI e entrada/saída de dados no modo I²C.
RC5/SDO	24	I/O (ST)	RC5 também pode ser saída de dados no modo SPI.
RC6/TX/CK	25	I/O (ST)	RC6 também pode ser TX (modo assíncrono - USART) ou clock (síncrono).
RC7/RX/DT	26	I/O (ST)	RC7 também pode ser RX (modo assíncrono - USART) ou dados (síncrono).
A Porta D é uma porta de entrada/saída bidirecional ou porta paralela escrava quando conectada a um barramento de microprocessador.			
RD0/PSP0	19	I/O (ST/TTL) [3]	
RD1/PSP1	20	I/O (ST/TTL) [3]	
RD2/PSP2	21	I/O (ST/TTL) [3]	
RD3/PSP3	22	I/O (ST/TTL) [3]	
RD4/PSP4	27	I/O (ST/TTL) [3]	
RD5/PSP5	28	I/O (ST/TTL) [3]	
RD6/PSP6	29	I/O (ST/TTL) [3]	
RD7/PSP7	30	I/O (ST/TTL) [3]	
A Porta E é uma porta de entrada/saída bidirecional.			
RE0/~~RD~~/AN5	8	I/O (ST/TTL) [3]	RE0 também pode ser controle de leitura p/ porta paralela ou entrada analógica 5.
RE1/~~WR~~/AN6	9	I/O (ST/TTL) [3]	RE1 também pode ser controle de escrita p/ porta paralela ou entrada analógica 6.
RE2/~~CS~~/AN7	10	I/O (ST/TTL) [3]	RE2 também pode ser controle de seleção p/porta paralela ou entrada analógica 7.

Legenda: I (entrada) O (saída) P (alimentação) TTL (entrada nível TTL) ST (entrada Schmidt Trigger) CMOS (entrada CMOS).
~~TACHADO~~ – ativado com nível lógico zero (0 Vcc).

Nota: 1 - Entrada ST quando configurada como uma interrupção externa.
2 - Entrada ST quando usada no modo de programação serial.
3 - Entrada ST quando configurada como entrada/saída de uso geral, e como TTL quando usada como porta paralela.
4 - Entrada ST quando configurada no modo oscilador RC e CMOS nos outros casos.

ANEXO II – PIC16F877: REGISTRADORES

Banco 0		Banco 1		Banco 2		Banco 3	
End.	**Descrição**	**End.**	**Descrição**	**End.**	**Descrição**	**End.**	**Descrição**
00h	*	80h	*	100h	*	180h	*
01h	TMR0	81h	OPTION_REG	101h	TMR0	181h	OPTION_REG
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PORTA	85h	TRISA	105h	-	185h	-
06h	PORTB	86h	TRISB	106h	PORTB	186h	TRISB
07h	PORTC	87h	TRISC	107h	-	187h	-
08h	PORTD	88h	TRISD	108h	-	188h	-
09h	PORTE	89h	TRISE	109h	-	189h	-
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh	INTCON
0Ch	PIR1	8Ch	PIE1	10Ch	EEDATA	18Ch	EECON1
0Dh	PIR2	8Dh	PIE2	10Dh	EEADR	18Dh	EECON2
0Eh	TMR1L	8Eh	PCON	10Eh	EEDATH	18Eh	Reservado[1]
0Fh	TMR1H	8Fh	-	10Fh	EEADRH	18Fh	Reservado[1]
10h	T1COM	90h	-	110h	16 bytes para registradores de uso geral	190h	16 bytes para registradores de uso geral
11h	TMR2	91h	SSPCON2	111h		191h	
12h	T2COM	92h	PR2	112h		192h	
13h	SSPBUF	93h	SSPADD	113h		193h	
14h	SSPCON	94h	SSPSTAT	114h		194h	
15h	CCPR1L	95h	-	115h		195h	
16h	CCPR1H	96h	-	116h		196h	
17h	CCP1COM	97h	-	117h		197h	
18h	RCSTA	98h	TXSTA	118h		198h	
19h	TXREG	99h	SPBRG	119h		199h	
1Ah	RCREG	9Ah	-	11Ah		19Ah	
1Bh	CCPR2L	9Bh	-	11Bh		19Bh	
1Ch	CCPR2H	9Ch	-	11Ch		19Ch	
1Dh	CCP2COM	9Dh	-	11Dh		19Dh	
1Eh	ADRESH	9Eh	ADRESL	11Eh		19Eh	
1Fh	ADCON0	9Fh	ADCON1	11Fh		19Fh	
20h a 7Fh	96 bytes para registradores de uso geral	A0h a EFh	80 bytes para registradores de uso geral	120h a 16Fh	80 bytes para registradores de uso geral	1A0h a 1EFh	80 bytes para registradores de uso geral
		F0h a FFh	Acessa 70h a 7Fh	170h a 17Fh	Acessa 70h a 7Fh	1F0h a 1FFh	Acessa 70h a 7Fh

Legenda: (*) Não é um registro fisicamente implementado, sendo usado para endereçamento indireto.
(-) Locações de memória de dados não implementadas, lidas como ‘0’ (zero).

Nota: 1 – Estes registros são reservados, mantenha-os limpos.

ANEXO III – PIC16F877: INSTRUÇÕES

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from Interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

ANEXO IV – M4088: REGISTRADORES

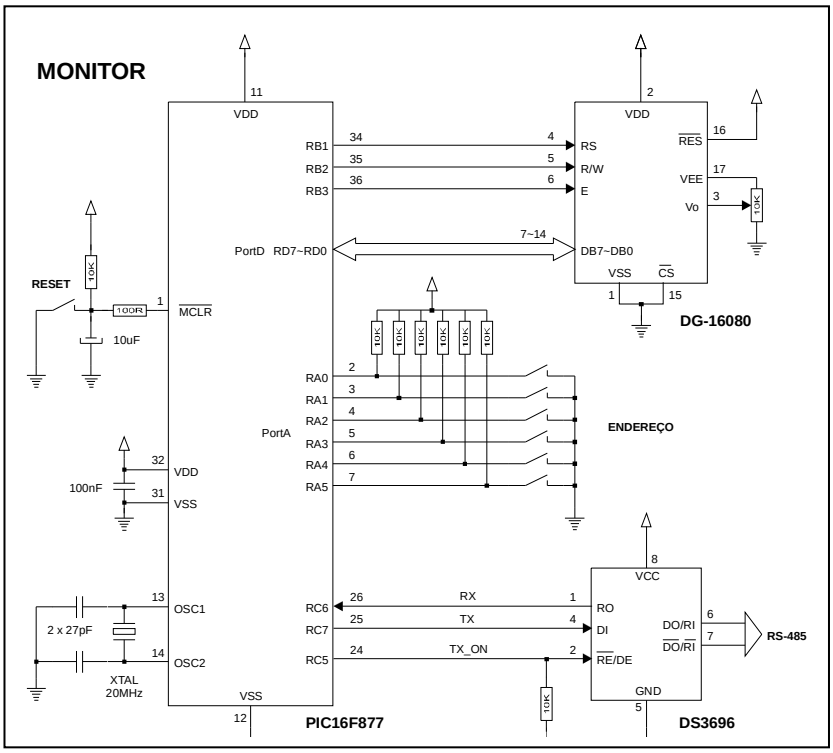
Registrador	Bit	Posição	Função
VPORT	VD7 – VD0	VPORT<7:0>	Seleciona a porta de dados de vídeo.
STATUS	TO2	STATUS<7>	Reservado.
	TO1	STATUS<6>	Reservado.
	OV	STATUS<3>	Flag de overrun de dados do pixel. É setado cada vez que um pixel é atualizado antes de RDY ser zerado. A leitura do registro zera o bit.
	VSYNC	STATUS<2>	Este bit duplica o sinal do pino VSYNC.
	HREF	STATUS<1>	Este bit duplica o sinal do pino HREF.
	RDY	STATUS<0>	Este bit é setado cada vez que um pixel é atualizado, e é zerado pela leitura do registro VPORT.
FCTL	FSET	FCTL<7>	Setado para iniciar a transferência de um frame único, mas somente se SFR estiver setado. Se for setado no meio de um frame, HREF não será mantido para o próximo frame. É zerado no final de cada frame.
	SFR	FCTL<6>	Habilita modo de operação em frame único. Em modo contínuo, a validade do dado é qualificada pela asserção de HREF a cada novo frame. No modo de frame único, HREF é afirmado somente para o primeiro frame imediatamente depois de setado FCTL<7>. A duração da afirmação de HREF é programada pelo tamanho da janela.
	SKIP	FCTL<3>	Faz VSYNC e HREF saltarem um frame. Esta função não altera a taxa de pixel, simplesmente bloqueia a afirmação em qualquer outro frame.
	FBLC	FCTL<2>	Seleciona quantas vezes a calibragem de nível escuro é internamente executada.
	STOP	FCTL<1>	Setado para entrar em modo standby (baixo consumo), parando o clock do chip. Esta função não altera o conteúdo dos registradores nem impede o acesso aos mesmos. O chip é colocado em modo default e todo o dado de imagem é perdido. Uma vez zerado o bit, normalmente aguarda-se dois frames para que o chip torne-se estável.
	SRST	FCTL<0>	Quando setado, promove a inicialização dos registradores internos, colocando o chip em estado default. Uma vez zerado o bit, normalmente aguarda-se dois frames para que o chip torne-se estável.
EXCTL	AUTO	EXCTL<7>	Habilita modo exposição automática. Zerar o bit para modo de exposição manual.
	EX	EXCTL<6:0>	Configura o tempo de exposição para 1/50s (7Fh) até 1/(50*128*2)s (00h). Válido somente para modo manual. Uma vez alterado o valor, aguarda-se dois frames para que o chip torne-se estável.
GCTL	GN	GCTL<2:0>	Configura o ganho de amplificação de 0dB (000h) a 18dB (111h) numa relação linear. Válido somente para modo manual. Uma vez alterado o valor, aguarda-se dois frames para que o chip torne-se estável.
FRCTL	FDIV	FRCTL<5:0>	Divisor de taxa de frame. Uma vez alterado o valor, aguarda-se dois frames para que o chip torne-se estável. Frame Rate = F0 / (FDIV+1) Pixel Rate = fosc / [(FDIV +1)*2]
MCTL	GAMA	MCTL<7>	Setar para gama = 0,45 e zerar para gama = 1.
	MIR	MCTL<6>	Setar para espelhar imagem.
	NSR	MCTL<5>	Setar para uso interno da câmera e zerar para uso externo.
	BKL	MCTL<4>	Setar para ligar compensação de luz de fundo.
	FZEX	MCTL<3>	Setar para manter setagem de exposição, somente em modo de exposição automática.
	PCKS	MCTL<2>	Zerar para gerar clock contínuo em PCLK. Setar para clock somente durante janela de pixel válida.
	PCKI	MCTL<1>	Setar para inverter sinal em PCLK.
	BPSHP	MCTL<0>	Setar para desabilitar função sharpness.
HWCTL/ VWCTL	HWS	HWCTL<7:4>	Início da janela - horizontal.
	HWE	HWCTL<3:0>	Fim da janela - horizontal.
	VWS	VWCTL<7:4>	Início da janela - vertical.
	VWE	VWCTL<3:0>	Fim da janela - vertical.

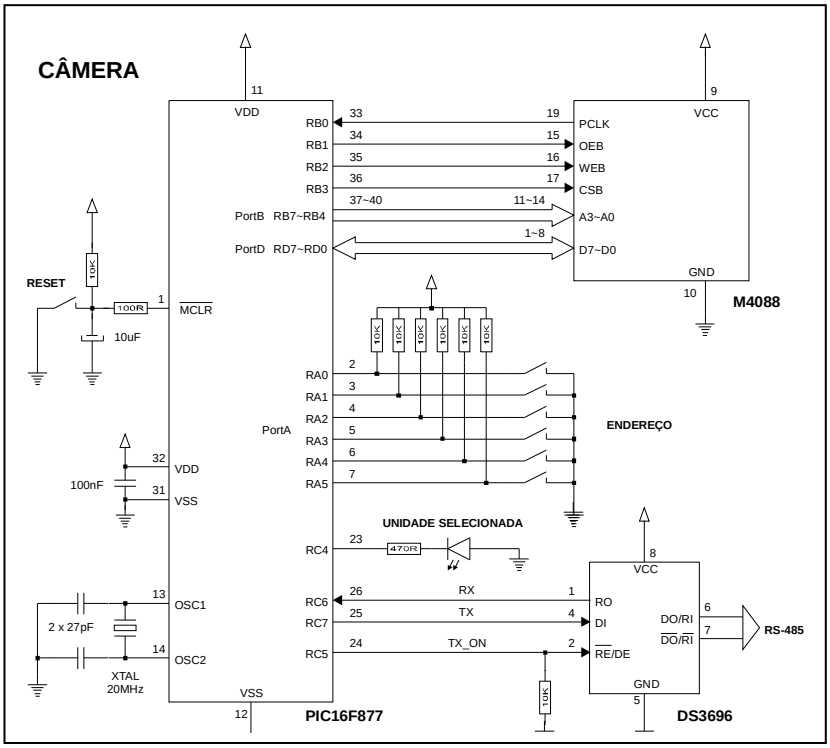
Nota: A imagem é dividida em blocos (16 x 16), e cada bloco com 24 pixels de altura por 18 de largura. Usa-se HWS, HWE, VWS e VWE para definir o tamanho da janela, definindo-se o endereço dos blocos inicial e final de cada direção. As características de definição de janela mudam somente o tempo de asserção de HREF, não influindo nas taxas de pixel ou dados. Se o endereço final for igual ou menor que o de início, então a janela vai do endereço de início até a borda mais a direita.

M4088: CONFIGURAÇÃO PADRÃO DOS REGISTRADORES

Registrador	Bit	Posição	Valor padrão	Configuração
VPORT	VD7 – VD0	VPORT<7:0>	XXXXXXXX	Indefinido.
STATUS	TO2	STATUS<7>	0	Reservado.
	TO1	STATUS<6>	0	Reservado.
	OV	STATUS<3>	X	Indefinido.
	VSYNC	STATUS<2>	X	Indefinido.
	HREF	STATUS<1>	X	Indefinido.
	RDY	STATUS<0>	X	Indefinido.
FCTL	FSET	FCTL<7>	0	Modo contínuo.
	SFR	FCTL<6>	0	Modo contínuo.
	SKIP	FCTL<3>	0	Modo contínuo.
	FBLC	FCTL<2>	0	Não executa teste de calibragem.
	STOP	FCTL<1>	0	Modo normal.
	SRST	FCTL<0>	0	Modo normal.
EXCTL	AUTO	EXCTL<7>	1	Modo de exposição automática.
	EX	EXCTL<6:0>	1111111	Configura o tempo de exposição para 1/50s (7Fh).
GCTL	GN	GCTL<2:0>	000	Configura o ganho de amplificação de 0dB (000h).
FRCTL	FDIV	FRCTL<5:0>	000000	Frame Rate = F0 Pixel Rate = fosc / 2] where fosc is the main clock frequency of XCLKi F0 = fosc / (458*625); F0=50 @ 14.318Mhz
MCTL	GAMA	MCTL<7>	0	Gama = 1.
	MIR	MCTL<6>	0	Imagem normal (não espelhada).
	NSR	MCTL<5>	0	Câmera externa.
	BKL	MCTL<4>	0	Sem compensação de luz de fundo.
	FZEX	MCTL<3>	0	Configuração flutuante.
	PCKS	MCTL<2>	0	Clock contínuo em PCLK.
	PCKI	MCTL<1>	0	Sinal PCLK normal.
	BPSHP	MCTL<0>	0	Função sharpness habilitada.
HWCTL / VWCTL	HWS	HWCTL<7:4>	0000	Tamanho máximo.
	HWE	HWCTL<3:0>	0000	Tamanho máximo.
	VWS	VWCTL<7:4>	0000	Tamanho máximo.
	VWE	VWCTL<3:0>	0000	Tamanho máximo.

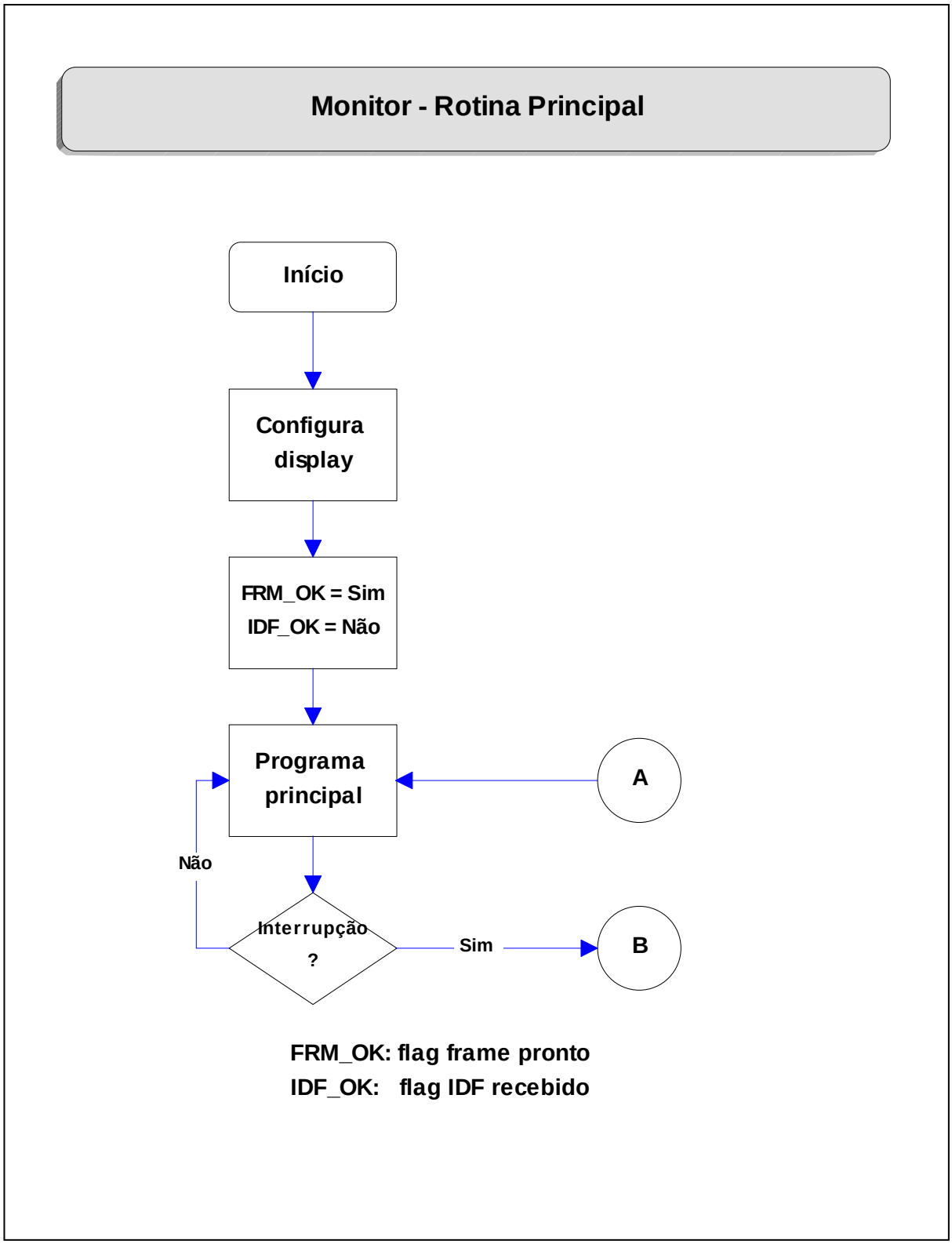
APÊNDICE I – O PROTÓTIPO: HARDWARE FINAL



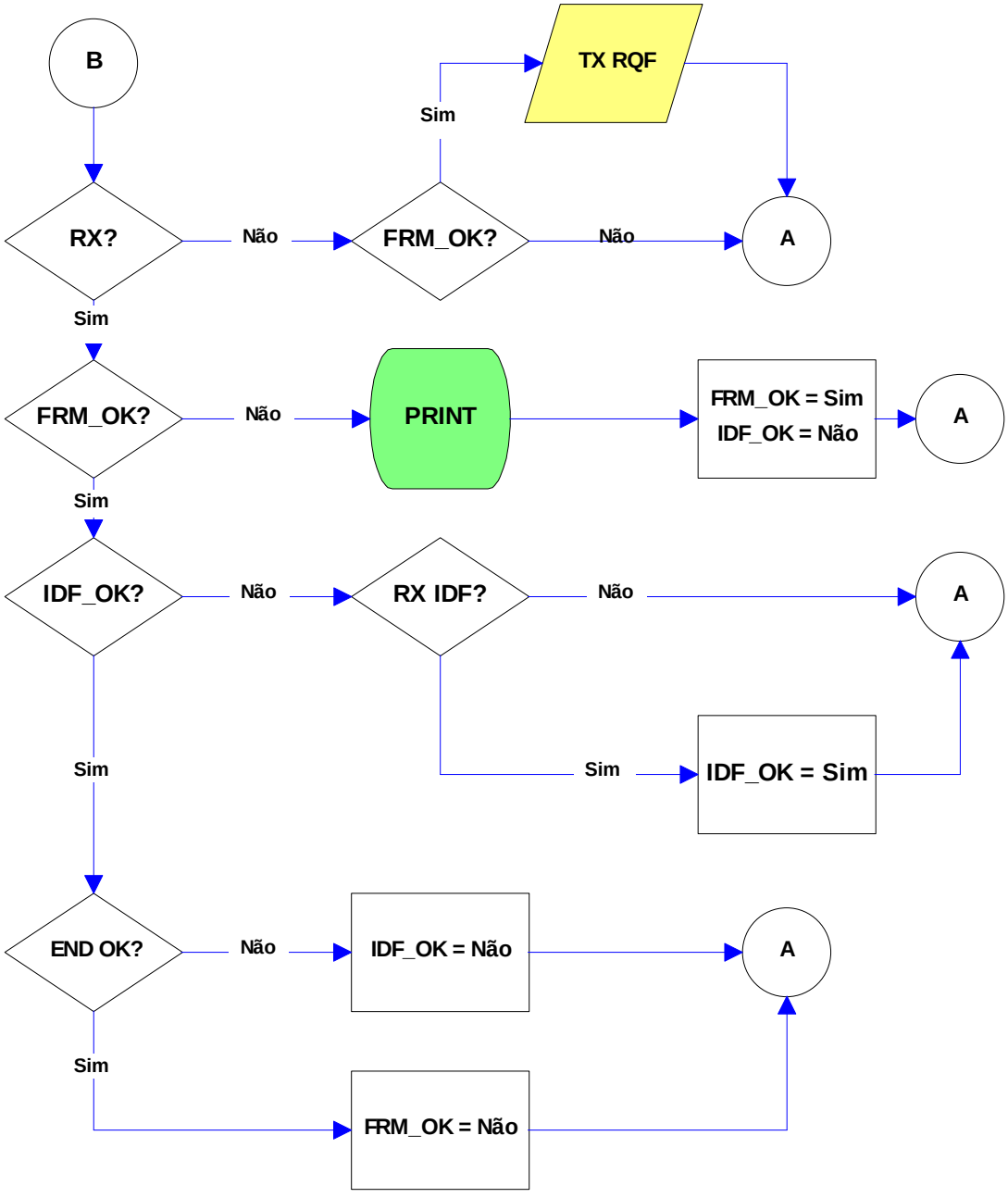


APÊNDICE II – O PROTÓTIPO: FLUXOGRAMAS

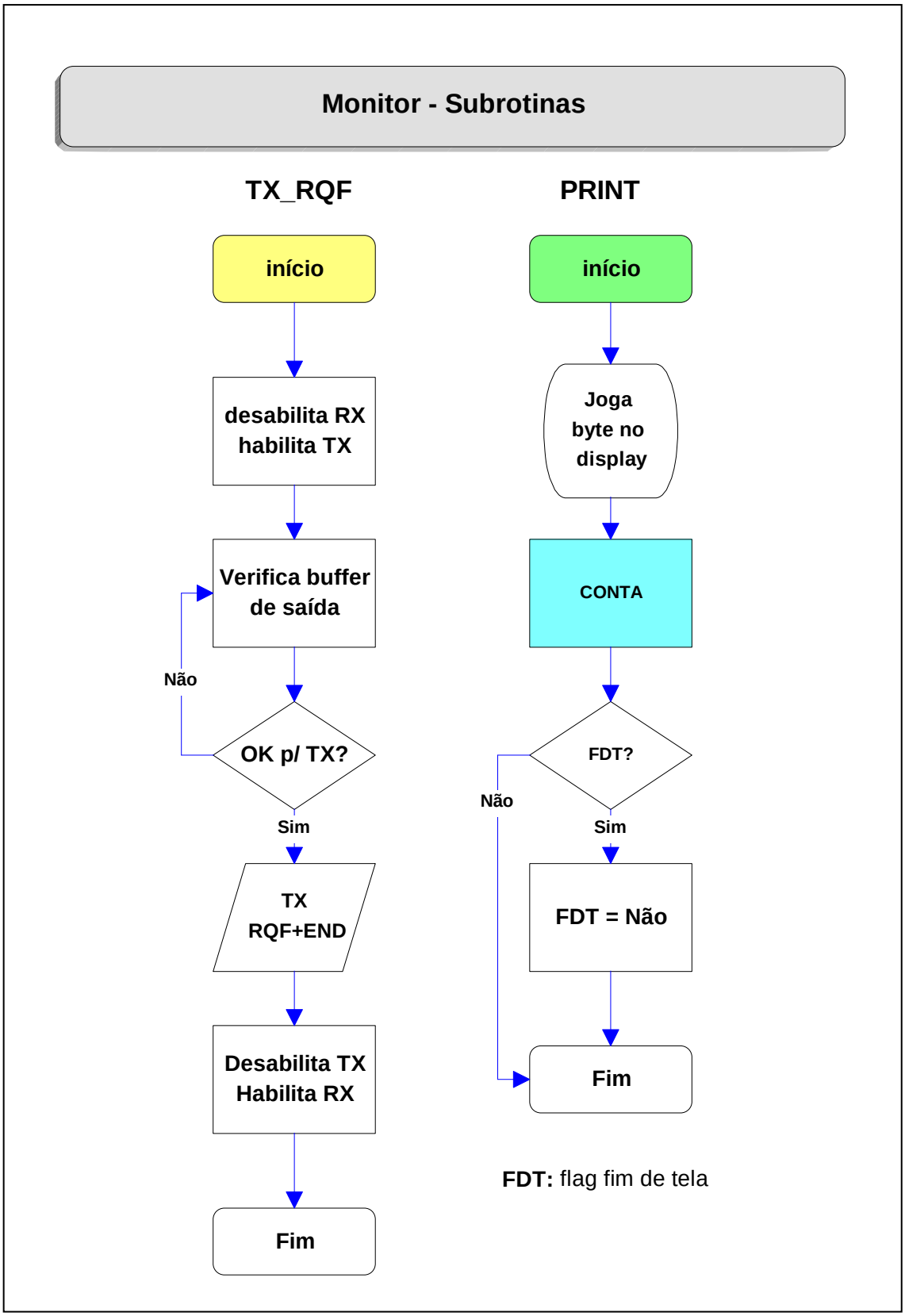
MONITOR

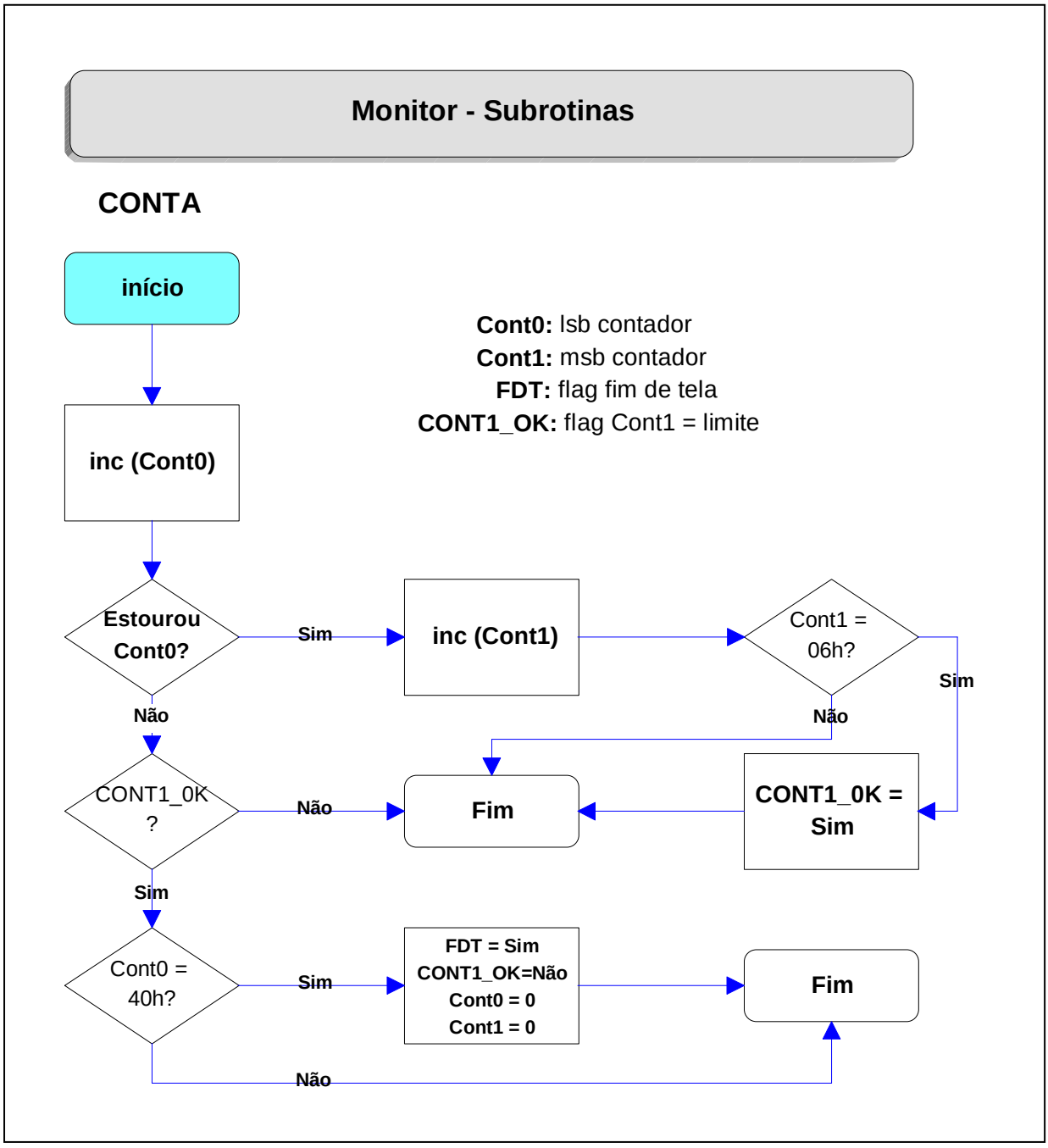

Monitor - Rotina Principal
Início
Configura display
FRM_OK = Sim
IDF_OK = Não
Programa principal
A
Não
Interrupção ?
Sim
B
FRM_OK: flag frame pronto
IDF_OK: flag IDF recebido

Monitor - Tratamento de Interrupção

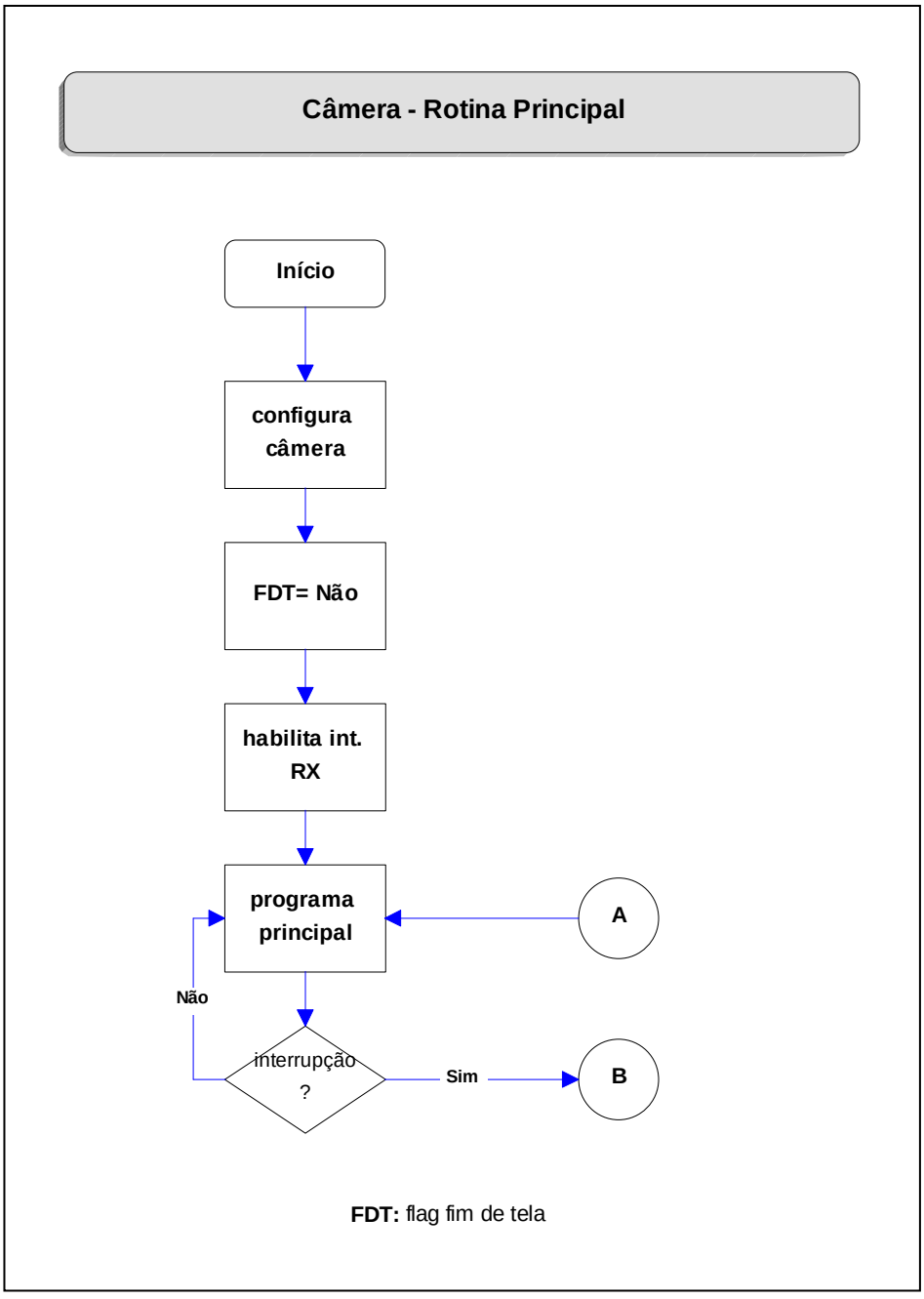


END: endereço
FRM_OK: flag frame pronto
IDF_OK: flag IDF recebido
PRINT: joga dado no display


Monitor - Subrotinas
TX_RQF
início
desabilita RX
habilita TX
Verifica buffer
de saída
Não
OK p/ TX?
Sim
TX
RQF+END
Desabilita TX
Habilita RX
Fim
PRINT
início
Joga
byte no
display
CONTA
FDT?
Não
Sim
FDT = Não
Fim
FDT: flag fim de tela

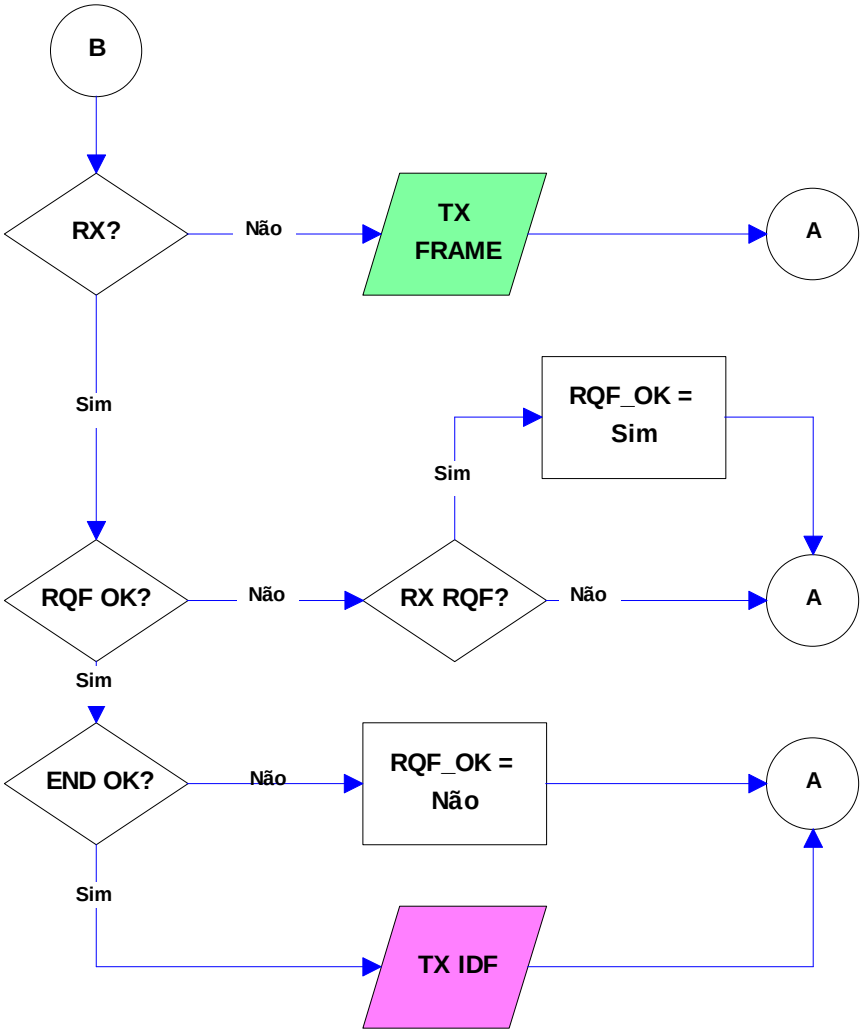

Monitor - Subrotinas
CONTA
início
inc (Cont0)
Estourou Cont0?
Sim
inc (Cont1)
Cont1 = 06h?
Sim
Não
Não
CONT1_0K ?
Não
Fim
CONT1_0K = Sim
Sim
Cont0 = 40h?
Sim
FDT = Sim
CONT1_OK=Não
Cont0 = 0
Cont1 = 0
Fim
Não
Cont0: lsb contador
Cont1: msb contador
FDT: flag fim de tela
CONT1_OK: flag Cont1 = limite

CÂMERA

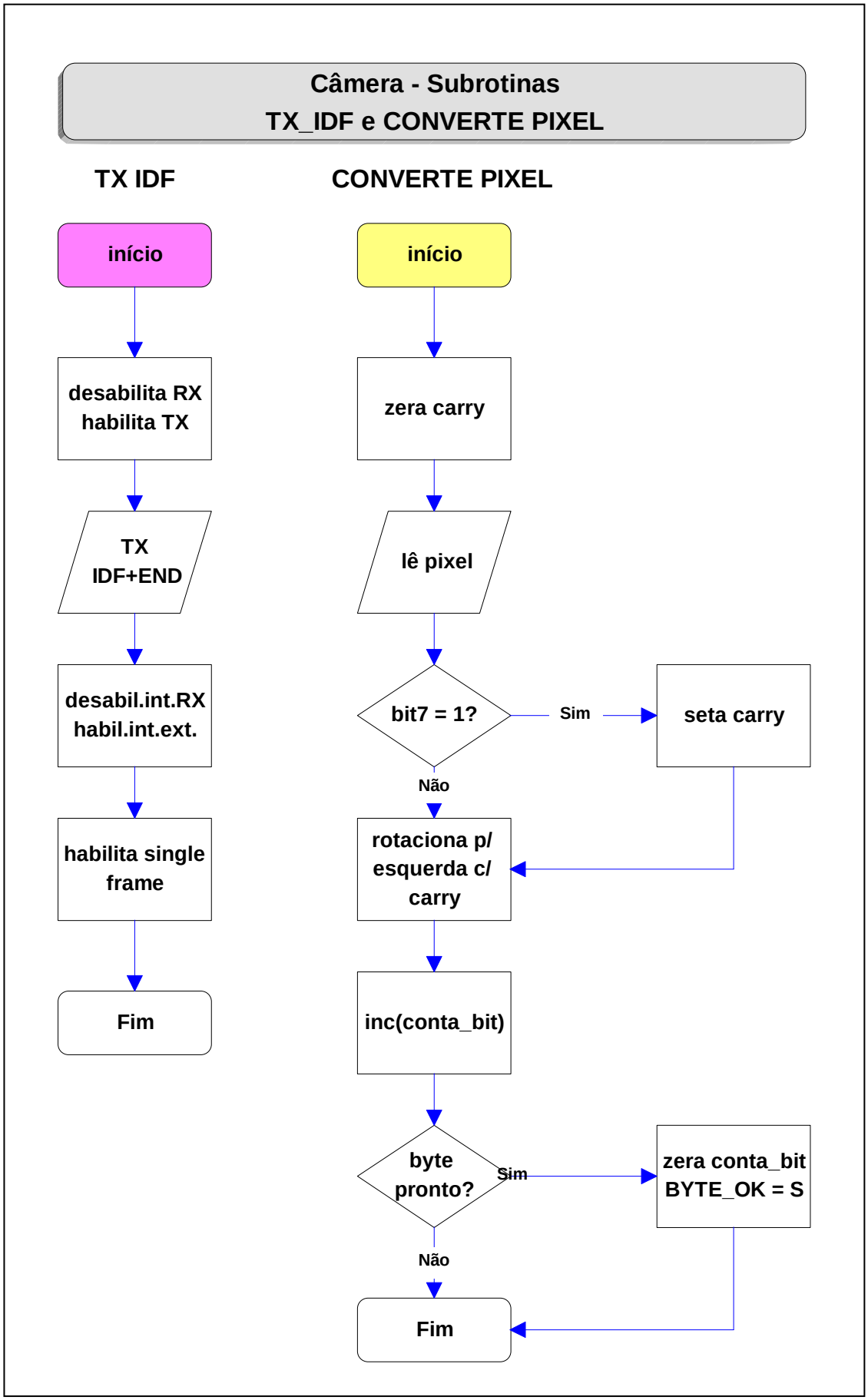


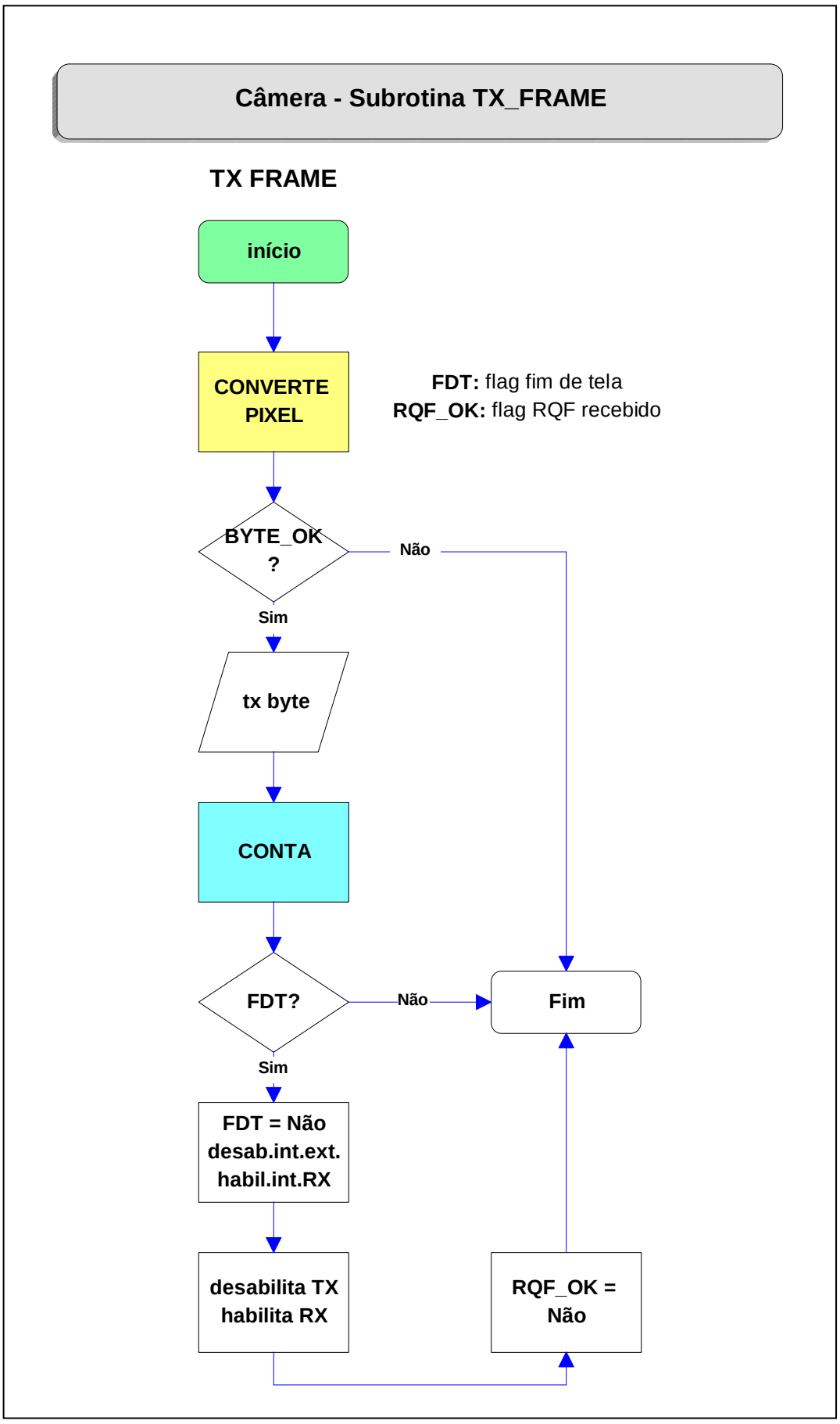
FDT: flag fim de tela

Câmera - Tratamento de Interrupção (modo single frame)



END: endereço
RQF_OK: flag RQF recebido


Câmera - Subrotinas
TX_IDF e CONVERTE PIXEL
TX IDF
CONVERTE PIXEL
início
desabilita RX
habilita TX
TX
IDF+END
desabil.int.RX
habil.int.ext.
habilita single
frame
Fim
início
zera carry
lê pixel
bit7 = 1?
Sim
seta carry
Não
rotaciona p/
esquerda c/
carry
inc(conta_bit)
byte
pronto?
Sim
zera conta_bit
BYTE_OK = S
Não
Fim


Câmera - Subrotina TX_FRAME
TX FRAME
início
CONVERTE PIXEL
FDT: flag fim de tela
RQF_OK: flag RQF recebido
BYTE_OK ?
Não
Sim
tx byte
CONTA
FDT?
Não
Fim
Sim
FDT = Não
desab.int.ext.
habil.int.RX
desabilita TX
habilita RX
RQF_OK = Não

APÊNDICE III – O PROTÓTIPO: SOFTWARE

MONITOR

```
;=====================================================================
; FURB - BCC - TCC - PROF. ANTONIO CARLOS TAVARES
;
;      AUTOR: Angelo Dias dos Santos
;      DATA: 24/06/2002
; FINALIDADE: rx dados via serial assincrona e jogar no display
;   OBJETIVO: - implementar monitor
;=====================================================================
;----------------------------- CONFIGURACOES INICIAIS -------------------
       LIST p=16F877         ;processador = 16F877
       INCLUDE<P16F877.INC>  ;arquivo com as definicoes do 16F877
                             ;configuracoes gerais do dispositivo:
       __CONFIG _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF &
_PWRTE_ON & _WDT_ON & _HS_OSC
;
;----------------------------- DEFINICAO DE FLAGS E CONSTANTES ----------
;
                             ;definicao de flags
#DEFINE IDF_OK  FLAGS,0      ; inicio de frame ok (cabecalho do frame)
                             ;  1 - idf rx
                             ;  0 - aguardando rx idf
#DEFINE FRM_OK  FLAGS,1      ; frame ok
                             ;  1 - frame rx
                             ;  0 - aguardando ou rx frame
#DEFINE FDT     FLAGS,2      ; fim de tela (lcd)
                             ;  1 - varredura completa
                             ;  0 - executando varredura
#DEFINE C1_OK   FLAGS,3      ; cont1 (msb contador)
                             ;  1 - igual a valor definido na subrotina
                             ;
                             ;definicao de constantes
IDF     EQU     B'00110011'  ; comando inicio de frame
RQF     EQU     B'01010101'  ; comando requisicao de frame
;
;----------------------------- DEFINICAO DE VARIAVEIS (mapa de memoria) -
;
FLAGS   EQU     H'20'        ;flags de uso geral
                             ; <7:> nao usados
                             ;   <3> c1_ok
                             ;   <2> fdt
                             ;   <1> frm_ok
                             ;   <0> idf_ok
DADO    EQU     H'21'        ;dado rx/tx
INST    EQU     H'22'        ;instrucao (configuracao/escrita do lcd)
OPER    EQU     H'23'        ;operando  (configuracao/escrita do lcd)
CONT0   EQU     H'24'        ;contador de bytes rx (lsb)
CONT1   EQU     H'25'        ;contador de bytes rx (msb)
LIM_C1  EQU     H'26'        ;limite p/ msb contador
LIM_C0  EQU     H'27'        ;limite p/ lsb contador
;
;----------------------------- DEFINICAO DE ENTRADAS --------------------
;
#DEFINE BSY     PORTD,7      ;flag lcd busy
                             ; 0 - ready (pronto)
                             ; 1 - busy (ocupado)
;
;----------------------------- DEFINICAO DE SAIDAS ----------------------
;
#DEFINE TX_ON   PORTC,5      ;chip interface serial 485 - pino 2
                             ; 1 - habilita tx
                             ; 0 - habilita rx
#DEFINE RS      PORTB,1      ;LCD Register Select
                             ; 0 - operando
                             ; 1 - instrucao
#DEFINE RW      PORTB,2      ;LCD Read /Write
                             ; 0 - escrita
                             ; 1 - leitura
#DEFINE E       PORTB,3      ;LCD Enable
                             ; 1 - executa instrucao
;
;----------------------------- VETOR DE RESET ---------------------------
;
       ORG     H'00'         ;endereco inicial do reset
       GOTO    INICIO        ;desvia p/ inicio do programa
```

```
;
;------------------------------ VETOR DE INTERRUPCAO ---------------------
;
       ORG     H'04'           ;end. inicial p/ tratamento de interrupcao
       BCF     STATUS,RP0      ; -> seleciona banco0
                               ;seleciona tipo de interrupcao (rx/tmr0):
       BTFSS   PIR1,RCIF       ; eh interrupcao de rx?
       GOTO    TIMEOUT         ;  nao, entao trata overflow de tmr0
                               ;le dado, trata se erro e salva dado rx:
       MOVF    RCREG,W         ; w <- rcreg = le dado rx
       BTFSS   RCSTA,OERR      ; houve sobrecarga de dados?
       GOTO    $+4             ;  nao, entao salta tratamento de erro
       BCF     RCSTA,CREN      ; limpa cren p/ zerar erro
       BSF     RCSTA,CREN      ; volta a habilitar rx continua
       MOVF    RCREG,W         ; le outro dado rx p/ esvaziar rcreg
       MOVWF   DADO            ; salva dado rx
                               ;confere protocolo de comunicacao:
       BTFSS   FRM_OK          ; frame rx?
       GOTO    PRINT           ;  nao, entao joga dado no display
       BTFSC   IDF_OK          ; idf rx?
       GOTO    RX_END          ;  sim, entao desvia p/ testar endereco rx
                               ; testa se dado recebido eh idf
       MOVF    DADO,W          ;  w <- dado rx
       XORLW   IDF             ;  w <- idf xor w (testa dado rx)
       BTFSC   STATUS,Z        ;  eh inicio de frame (idf)?
       BSF     IDF_OK          ;   sim, entao seta flag
       RETFIE                  ;retorna ao programa principal
;
RX_END:                        ;testa se endereco rx = portA
       MOVF    DADO,W          ; w <- dado rx
       XORWF   PORTA,W         ; w <- portA xor w (testa endereco rx)
       BTFSS   STATUS,Z        ; endereco confere?
       GOTO    $+3             ;  nao, salta p/ tratar erro
       BCF     FRM_OK          ; zera flag p/ rx frame
       RETFIE                  ;retorna ao programa principal
       BCF     IDF_OK          ; erro de end: zera flag p/ tx novo rqf
       RETFIE                  ;retorna ao programa principal
;
TIMEOUT:                       ;trata estouro de tmr0 - timeout
       BTFSC   FRM_OK          ; frame rx?
       CALL    TX_RQF          ;  sim, entao tx requisicao p/ novo frame
       BCF     INTCON,T0IF     ; limpa t0if p/ nova interrupcao de tmr0
       RETFIE                  ;retorna ao programa principal
;
PRINT:                         ;joga dado rx no display
       CLRWDT                  ;zera watchdog
       MOVLW   H'0C'           ; w <- 0Ch (endereco RAM de video)
       MOVWF   INST            ; codigo instrucao
       MOVF    DADO,W          ; w <- dado rx
       MOVWF   OPER            ; operando (dado a ser "impresso" no lcd)
       CALL    EXELCD          ; joga instrucao/operando
       CALL    CONTA           ; incrementa contador
       BTFSS   FDT             ; eh fim de tela?
       RETFIE                  ;retorna ao programa principal
       BCF     FDT             ; zera flag p/ nova varredura do display
       CALL    PROMPT          ; reposiciona cursor no inicio da tela
       BSF     FRM_OK          ; seta flag p/ rx nova rqf
       BCF     IDF_OK          ; zera flag p/ rx novo idf
       RETFIE                  ;retorna ao programa principal
;
;------------------------------ INICIO DO PROGRAMA ----------------------
;
INICIO:
       BCF     STATUS,RP1      ;-> garante selecao banco0
       BCF     STATUS,RP0      ;-> garante selecao banco0
                               ;
                               ;inicializacoes:
       CLRF    PORTA           ; limpa portA
       CLRF    PORTB           ; limpa portB
       CLRF    PORTC           ; limpa portC
       CLRF    PORTD           ; limpa portD
       CLRF    PORTE           ; limpa portE
       CLRF    FLAGS           ; limpa flags
       CLRF    CONT0           ; limpa lsb contador
```

```
CLRF    CONT1           ; limpa msb contador
BSF     FRM_OK          ; seta flag p/ forcar primeira tx de rqf
                        ; lim_c0 e lim_c1 sao definidos em clrlcd
                        ;
                        ;configura portas de entrada/saida:
BSF     STATUS,RP0      ; -> seleciona banco1
                        ;portA: entrada de endereco
MOVLW   B'00000110'     ; configura todos os pinos do portA...
MOVWF   ADCON1          ; ...como I/O digital.
MOVLW   B'00111111'     ; mascara p/ configuracao do portA
MOVWF   TRISA           ; portA <7:6>nao disponiveis <5:0>entradas
                        ;portB: sinais de controle do display
MOVLW   B'00000000'     ; mascara p/ configuracao do portB
MOVWF   TRISB           ; portB <7:0>saidas
                        ;portC: controle e rx/tx chip serial
MOVLW   B'11000000'     ; mascara p/ configuracao do portC
MOVWF   TRISC           ; portC <7:6>entradas <5:0>saidas
                        ;portD: barramento de dados do display
BCF     TRISE,PSPMODE   ; configura portD como I/O generico
MOVLW   B'00000000'     ; mascara p/ configuracao do portD
MOVWF   TRISD           ; portD <7:0>saidas
BCF     STATUS,RP0      ; -> seleciona banco0
                        ;
                        ;configura display:
                        ; qtde bytes por linha
MOVLW   H'02'           ;  w    <- 02h (reg. numero caracteres)
MOVWF   INST            ;  INST <- w
MOVLW   D'17'           ;  w    <- 17 (18 bytes/linha)
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ; divisao de tempo - refresh
MOVLW   H'03'           ;  w    <- 03h (registro divisoes tempo)
MOVWF   INST            ;  INST <- w
MOVLW   D'79'           ;  w    <- 79?? (64 PARECE ACEITAVEL)
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ; qtde bits por palavra (tamanho do byte)
MOVLW   H'01'           ;  w    <- 01h (registro tamanho caracter)
MOVWF   INST            ;  INST <- w
MOVLW   H'77'           ;  w    <- 77h (07h - 8 bits por byte)
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ; modo gráfico
MOVLW   H'00'           ;  w    <- 00h (registro controle de modo)
MOVWF   INST            ;  INST <- w
MOVLW   H'32'           ;  w    <- 32h (modo grafico - master)
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ; endereco inicial display (low)
MOVLW   H'08'           ;  w    <- 08h (reg. endereco inicial LOW)
MOVWF   INST            ;  INST <- w
MOVLW   H'00'           ;  w    <- 00h
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ; endereco inicial display (high)
MOVLW   H'09'           ;  w    <- 09h (reg. endereco inicial HIGH)
MOVWF   INST            ;  INST <- w
MOVLW   H'00'           ;  w    <- 00h
MOVWF   OPER            ;  OPER <- w
CALL    EXELCD          ;  executa operacao no lcd
                        ;
CALL    CLRLCD          ; inicializa (limpa) display
                        ;
                        ;configura usart:
BSF     STATUS,RP0      ; -> seleciona banco1
CLRF    SPBRG           ; configura taxa de tx em 1,25Mbps
BSF     TXSTA,BRGH      ; configura baud rate em modo rapido
BCF     TXSTA,SYNC      ; configura modo assincrono
BSF     TXSTA,TXEN      ; habilita transmissao
BCF     STATUS,RP0      ; -> seleciona banco0
BCF     RCSTA,CREN      ; desabilita recepcao continua
BCF     RCSTA,SREN      ; desabilita recepcao single
BSF     RCSTA,SPEN      ; habilita porta serial (pinos RC6 e RC7)
BCF     TX_ON           ; habilita rx chip interface serial
```

```
                            ;
                            ;configura interrupcao:
        MOVLW   B'11100000' ; w recebe configuracao p/ intcon
        MOVWF   INTCON      ; habilita interrupcao de tmr0 e de periferico
        BSF     STATUS,RP0  ; -> seleciona banco1
        MOVLW   B'00000111' ; w recebe configuracao p/ option_reg
        MOVWF   OPTION_REG  ; prescaler em 1:256 p/ tmr0
        BSF     PIE1,RCIE   ; habilita interrupcao de rx
        BCF     STATUS,RP0  ; -> seleciona banco0
;
;------------------------------ ROTINA PRINCIPAL -------------------------
;
MAIN:
        GOTO    MAIN        ;aguardando rx ou timeout.
;
;------------------------------ SUB-ROTINAS GENERICAS --------------------
;
CONTA:                      ;contador (limites definidos por constantes)
        INCFSZ  CONT0,F     ; incrementa lsb e testa: igual a 0?
        GOTO    $+2         ;  nao, salta p/ testar cont1 e cont0
        GOTO    INC_C1      ;  sim, incrementa cont1
        BTFSS   C1_OK       ; cont1 = lim_c1?
        RETURN              ;  nao, encerra rotina.
        MOVF    LIM_C0,W    ;  sim, w recebe limite de cont0
        XORWF   CONT0,W     ; compara contador 0 c/ valor limite
        BTFSS   STATUS,Z    ; cont0 = lim_c0?
        RETURN              ;  nao, encerra rotina.
                            ; sim, cont1 = lim_c1 e cont0 = lim_c0
        BSF     FDT         ; seta flag fim de tela
        BCF     C1_OK       ; zera flag (cont1 = lim_c1)
        CLRF    CONT0       ; limpa lsb contador
        CLRF    CONT1       ; limpa msb contador
        RETURN              ;encerra rotina.
INC_C1                      ; incrementa cont1
        INCF    CONT1,F     ; incrementa byte msb do contador
        MOVF    LIM_C1,W    ;  sim, w recebe limite de cont1
        XORWF   CONT1,W     ; compara contador 1 c/ valor limite
        BTFSS   STATUS,Z    ;  cont1 = lim_c1?
        RETURN              ;   nao, entao encerra rotina.
        BSF     C1_OK       ;   sim, seta flag (cont1 = lim_c1)
        RETURN              ; encerra rotina.
;
;------------------------------ SUB-ROTINAS DISPLAY ----------------------
;
EXELCD:                     ;joga instrucao/operando no lcd p/ execucao
                            ;escreve instrucao
        CALL    LE_FLAG     ;testa lcd busy
        MOVF    INST,W      ; W <- INST
        MOVWF   PORTD       ; escreve codigo da instrucao
        BCF     RW          ; zera RW - escrita
        BSF     RS          ; seta RS - instrucao
        BSF     E           ; seta E - habilita operacao escrita
        NOP                 ; delay p/ lcd assimilar comando
        BCF     E           ; zera E - aguarda proxima operacao
                            ;escreve operando
        CALL    LE_FLAG     ; testa lcd busy
        MOVF    OPER,W      ; W <- OPER
        MOVWF   PORTD       ; escreve operando referente a instrucao
        BCF     RW          ; zera RW - escrita
        BCF     RS          ; zera RS - operando
        BSF     E           ; seta E - habilita operacao escrita
        NOP                 ; delay p/ lcd assimilar comando
        BCF     E           ; zera E - aguarda proxima operacao
        RETURN              ;encerra rotina
;
LE_FLAG:                    ;le flag lcd busy
        BSF     STATUS,RP0  ; -> seleciona banco1
        BSF     TRISD,7     ; configura bit7 do portD como entrada
        BCF     STATUS,RP0  ; -> seleciona banco0
BUSY:   BCF     E           ; zera E - aguarda proxima operacao
        BSF     RS          ; seta RS - instrucao
        BSF     RW          ; seta RW - leitura
        BSF     E           ; seta E - habilita operacao leitura
        BTFSC   BSY         ; testa flag lcd busy - lcd ready?
```

```
        GOTO    BUSY            ;  nao, loop enquanto lcd busy
        BCF     E               ; zera E - aguarda proxima operacao
        BSF     STATUS,RP0      ; -> seleciona banco1
        BCF     TRISD,7         ; configura bit7 do portD como saida
        BCF     STATUS,RP0      ; -> seleciona banco0
        RETURN                  ; encerra rotina
;
CLRLCD:                         ;limpa tela do display
        CALL    PROMPT          ; posiciona cursor
        MOVLW   H'0C'           ; w <- 0Ch (RAM)
        MOVWF   INST            ;
        MOVLW   B'00000000'     ; w <- 00h
        MOVWF   OPER            ;
        MOVLW   H'06'           ; w <- valor p/ limite c1
        MOVWF   LIM_C1          ; lim_c1 <- valor p/ varrer todo lcd
        MOVLW   H'40'           ; w <- valor p/ limite c0
        MOVWF   LIM_C0          ; lim_c0 <- valor p/ varrer todo lcd
LIMPA:                          ;escreve H'00' em todas posicoes do display
        CALL    EXELCD          ; executa instrucao
        CALL    CONTA           ; incrementa contador
        BTFSS   FDT             ; eh fim de tela?
        GOTO    LIMPA           ;  nao, limpa outro pixel
        BCF     FDT             ; limpa flag fim de tela p/ nova varredura
                                ;reinicializa limites p/ tela 144 x 72
        MOVLW   H'05'           ; w <- valor p/ limite c1
        MOVWF   LIM_C1          ; lim_c1 <- valor p/ compatibilizar cam/lcd
        MOVLW   H'10'           ; w <- valor p/ limite c0
        MOVWF   LIM_C0          ; lim_c0 <- valor p/ compatibilizar cam/lcd
        RETURN                  ;encerra rotina
;
PROMPT:                         ;posiciona cursor no canto superior esquerdo
                                ; posicao inicial cursor (low)
        MOVLW   H'0A'           ;  w    <- 0Ah (reg. contador endereco LOW)
        MOVWF   INST            ;  INST <- w
        MOVLW   H'00'           ;  w    <- 00h
        MOVWF   OPER            ;  OPER <- w
        CALL    EXELCD          ;  executa instrucao
                                ; posicao inicial cursor (high)
        MOVLW   H'0B'           ;  w    <- 0Bh (reg. contador endereco HIGH)
        MOVWF   INST            ;  INST <- w
        MOVLW   H'00'           ;  w    <- 00h
        MOVWF   OPER            ;  OPER <- w
        CALL    EXELCD          ;  executa instrucao
        RETURN                  ;encerra rotina
;
;------------------------------ SUB-ROTINAS USART -----------------------
;
TX_RQF:                         ;tx requisicao de frame
        BCF     RCSTA,CREN      ; desabilita recepcao
        BSF     TX_ON           ; habilita chip serial 485 a tx
                                ; tx comando:
        MOVLW   RQF             ;  w recebe comando a ser tx
        MOVWF   DADO            ;  salva dado a ser tx
        CALL    TX              ;  chama rotina p/ tx
                                ; tx endereco:
        MOVFW   PORTA           ;  w recebe endereco a ser tx
        MOVWF   DADO            ;  salva dado a ser tx
        CALL    TX              ;  chama rotina p/ tx
        BSF     RCSTA,CREN      ; habilita recepcao
        BCF     TX_ON           ; habilita rx chip interface serial
        RETURN                  ; encerra rotina
;
TX:                             ;tx dado
        BSF     STATUS,RP0      ; -> seleciona banco1
        BTFSS   TXSTA,TRMT      ; testa se buffer estah vazio
        GOTO    TX              ; loop enquanto buffer cheio
        BCF     STATUS,RP0      ; -> seleciona banco0
        MOVF    DADO,W          ; w recebe dado a ser tx
        MOVWF   TXREG           ; tx dado
DELAY_TX:                       ; aguarda buffer vazio p/ prosseguir
        BSF     STATUS,RP0      ; -> seleciona banco1
        BTFSS   TXSTA,TRMT      ;  testa se buffer estah vazio
        GOTO    DELAY_TX        ;  loop enquanto buffer cheio
        BCF     STATUS,RP0      ; -> seleciona banco0
```

```
        RETURN                  ; encerra rotina
;
;------------------------------ FIM DO PROGRAMA FONTE --------------------

        END                     ;obrigatorio
```

CÂMERA

```
;=====================================================================
; FURB - BCC - TCC - PROF. ANTONIO CARLOS TAVARES
;
;      AUTOR: Angelo Dias dos Santos
;       DATA: 25/06/2002
; FINALIDADE: Capturar imagem e tx via serial assincrona
;  OBJETIVOS: - implementar camera
;             - ADAPTAÇÃO PARA TESTE DE TX SEM CONTADOR (reducao de codigo)
;=====================================================================
;------------------------------ CONFIGURACOES INICIAIS -------------------
       LIST p=16F877          ;processador = 16F877
       INCLUDE<P16F877.INC>  ;arquivo com as definicoes do 16F877
                              ;configuracoes gerais do dispositivo:
       __CONFIG _CP_OFF & _DEBUG_OFF & _WRT_ENABLE_OFF & _CPD_OFF & _LVP_OFF & _BODEN_OFF &
_PWRTE_ON & _WDT_OFF & _HS_OSC
;
;------------------------------ DEFINICAO DE CONSTANTES ------------------
;
#DEFINE RQF_OK  FLAGS,0        ;requisicao de frame ok
                               ; 1 - rqf rx
                               ; 0 - aguardando rx rqf
;#DEFINE       FRM_OK  FLAGS,1 ;frame ok -> RESERVADO P/ MONITOR
                               ; 1 - frame tx
                               ; 0 - aguardando ou tx frame
#DEFINE FDT     FLAGS,2        ;fim de tela (lcd)
                               ; 1 - varredura completa
                               ; 0 - executando varredura
#DEFINE C1_OK   FLAGS,3        ;cont1 (msb contador)
                               ; 1 - igual a valor definido na subrotina
#DEFINE TX_PIX  FLAGS,7        ;selecao de pixel lido
                               ;
IDF     EQU     B'00110011'    ;comando inicio de frame
RQF     EQU     B'01010101'    ;comando requisicao de frame
;
;------------------------------ DEFINICAO DE VARIAVEIS (mapa de memoria) -
;
FLAGS   EQU     H'20'          ;flags de uso geral
                               ; <7:4> nao usados
                               ;   <3> c1_ok
                               ;   <2> fdt
                               ;   <1> frm_ok
                               ;   <0> rqf_ok
DADO    EQU     H'21'          ;dado rx/tx
BYTE_C  EQU     H'22'          ;byte a ser convertido
CONT    EQU     H'23'          ;contador de bit convertido
;
;------------------------------ DEFINICAO DE ENTRADAS --------------------
;
;------------------------------ DEFINICAO DE SAIDAS ----------------------
;
#DEFINE TX_ON  PORTC,5         ;chip interface serial 485 - pino 2
                               ; 1 - habilita tx
                               ; 0 - habilita rx
#DEFINE UNIT_ON PORTC,4        ;indica unidade selecionada
                               ; 1 - unidade tx frame
                               ;
                               ;controle do display
#DEFINE OEB    PORTB,1         ; OEB (Output Enable Bit)
                               ;  1 - barramento de dados em tristate
                               ;  0 - barramento habilitado
#DEFINE WEB    PORTB,2         ; WEB (Write Enable Bit)
                               ;  confirma escrita na borda de descida
#DEFINE CSB    PORTB,3         ; WEB (Write Enable Bit)
                               ;  0 - dispositivo selecionado
                               ; endereco p/ registro interno
#DEFINE A0     PORTB,4         ; A<0>
#DEFINE A1     PORTB,5         ; A<1>
#DEFINE A2     PORTB,6         ; A<2>
#DEFINE A3     PORTB,7         ; A<3>
;
;------------------------------ VETOR DE RESET ---------------------------
;
       ORG     H'00'          ;endereco inicial do reset
       GOTO    INICIO         ;desvia p/ inicio do programa
```

```
;
;------------------------------ VETOR DE INTERRUPCAO ---------------------
;
        ORG     H'04'           ;end. inicial p/ tratamento de interrupcao
        BCF     STATUS,RP0      ; -> seleciona banco0
                                ;confirma fonte interrupcao:
        BTFSC   INTCON,INTF     ; eh interrupcao externa?
        GOTO    INT_EXT         ;  sim, entao trata int.ext.
        BTFSC   PIR1,RCIF       ; eh interrupcao de rx?
        GOTO    RX              ;  sim, trata rx
                                ; entao trata overflow de tmr0
        BSF     RCSTA,CREN      ;  habilita recepcao
        BCF     TX_ON           ;  habilita rx chip interface serial
        BCF     UNIT_ON         ;  sinaliza unidade aguardando
        BCF     INTCON,T0IF     ;  limpa t0if p/ nova interrupcao de tmr0
        BCF     INTCON,T0IE     ;  habilita interrupcao tmr0
        RETFIE                  ;retorna ao programa principal
;
INT_EXT:                        ;trata int.ext (pclk)
        CLRF    TMR0            ; zera tmr0
        BTFSS   TX_PIX          ; tx byte?
        GOTO    SETA_TX         ;  nao, enta sai da rotina tratadora
                                ; le pixel no portD
        BCF     TX_PIX          ; zera flag p/ rx
        MOVF    PORTD           ; w recebe byte lido
        ANDLW   B'10000000'     ; compara bit7
        BSF     STATUS,C        ; seta carry p/ rotacionar (logica invertida)
        BTFSS   STATUS,Z        ; bit6 ligado?
        BCF     STATUS,C        ;  sim, seta carry p/ rotacionar
        RLF     BYTE_C,F        ; rotaciona bit de dado p/ esquerda
        INCF    CONT,F          ; incrementa contador
        BTFSS   CONT,3          ; byte montado? (testa bit ordem 3)
        RETFIE                  ; retorna ao programa principal
                                ;tx byte
        MOVF    BYTE_C,W        ; move byte convertido para w
        MOVWF   DADO            ; salva dado a ser tx
        CALL    TX              ; chama rotina p/ tx
        CLRF    CONT            ; limpa contador
        BCF     INTCON,INTF     ; zera flag p/ nova interrupcao
        RETFIE                  ; retorna ao programa principal
                                ;
SETA_TX:                        ; prepara p/ nova int.ext. em rb0
        BSF     TX_PIX          ;
        BCF     INTCON,INTF     ;  zera flag p/ nova interrupcao
        RETFIE                  ; retorna ao programa principal
;
RX:                             ;le dado, trata se erro e salva dado rx:
        MOVF    RCREG,W         ; w <- rcreg = le dado rx
        BTFSS   RCSTA,OERR      ; houve sobrecarga de dados?
        GOTO    $+4             ;  nao, entao salta tratamento de erro
        BCF     RCSTA,CREN      ; limpa cren p/ zerar erro
        BSF     RCSTA,CREN      ; volta a habilitar rx continua
        MOVF    RCREG,W         ; le outro dado rx p/ esvaziar rcreg
        MOVWF   DADO            ; salva dado rx
                                ;confere protocolo de comunicacao:
        BTFSC   RQF_OK          ; rqf rx?
        GOTO    RX_END          ;  sim, entao testa se endereco ok
                                ; testa se eh requisicao de frame
        MOVF    DADO,W          ; w <- dado rx
        XORLW   RQF             ; w <- rqf xor w (testa dado rx)
        BTFSC   STATUS,Z        ; eh requisicao de frame (rqf)?
        BSF     RQF_OK          ;  sim, entao seta flag
        RETFIE                  ;retorna ao programa principal
;
RX_END:                         ;testa se endereco rx = portA
        MOVF    DADO,W          ; w <- dado rx
        XORWF   PORTA,W         ; w <- portA xor w (testa endereco rx)
        BTFSC   STATUS,Z        ; endereco confere?
        GOTO    $+3             ;  sim, salta p/ tx frame
        BCF     RQF_OK          ; erro de end: zera flag p/ rx novo rqf
        RETFIE                  ; retorna ao programa principal
                                ; endereco ok, modo tx
        BCF     RQF_OK          ;  zera flag p/ rx novo rqf
        BSF     UNIT_ON         ;  sinaliza unidade selecionada
```

```
        CALL    TX_IDF          ;  chama rotina p/ tx comando idf+end
        BSF     INTCON,T0IE     ;  habilita interrupcao tmr0
        CALL    FSET            ;  dispara 1 frame
        CALL    VPORT           ;  habilita leitura da camer
        RETFIE                  ;retorna ao programa principal
;
;------------------------------ INICIO DO PROGRAMA ----------------------
;
INICIO:
        BCF     STATUS,RP1      ;-> garante selecao banco0
        BCF     STATUS,RP0      ;-> garante selecao banco0
                                ;
                                ;inicializacoes
        CLRF    INTCON          ; desabilita ints.p/garantir configs.
        CLRF    PORTA           ; limpa portA
        CLRF    PORTB           ; limpa portB
        CLRF    PORTC           ; limpa portC
        CLRF    PORTD           ; limpa portD
        CLRF    PORTE           ; limpa portE
        CLRF    FLAGS           ; limpa flags
        CLRF    CONT            ; limpa contador de bits convertidos
        BSF     TX_PIX          ; seta flag p/ tx primeiro pixel lido
                                ;
                                ;configura portas de entrada/saida:
        BSF     STATUS,RP0      ; -> seleciona banco1
                                ;portA: entrada de endereco
        MOVLW   B'00000110'     ; configura todos os pinos do portA...
        MOVWF   ADCON1          ; ...como I/O digital.
        MOVLW   B'00111111'     ; mascara p/ configuracao do portA
        MOVWF   TRISA           ; portA <7:6>nao disponiveis <5:0>entradas
                                ;portB: barramento de controle da camera
        MOVLW   B'00000001'     ; mascara p/ configuracao do portB
        MOVWF   TRISB           ; portB <7:1>saidas <0>entrada
        BCF     OPTION_REG,7    ; habilita pull-ups p/ portB
                                ;portC: controle e rx/tx chip serial
        MOVLW   B'11000000'     ; mascara p/ configuracao do portC
        MOVWF   TRISC           ; portC <7:6>entradas <5:0>saidas
                                ;portD: barramento de dados da camera
        BCF     TRISE,PSPMODE   ; configura portD como I/O generico
        MOVLW   B'00000000'     ; mascara p/ configuracao do portD
        MOVWF   TRISD           ; portD <7:0>inicialmente como saidas
        BCF     STATUS,RP0      ; -> seleciona banco0
                                ;
                                ;configura camera:
        BCF     CSB             ; inicializa CSB - dispositivo habilitado
        BSF     OEB             ; inicializa OEB - data bus em tristate
        CALL    SFR             ; habilita modo single frame
        CALL    PCKS            ; habilita pclk somente p/ pixel valido
        CALL    FPS             ; configura taxa de frames por segundo
        CALL    HWCTL           ; configura limite horizontal do frame
        CALL    VWCTL           ; configura limite vertical do frame
        CALL    FSET            ; executa ciclo inicial
        CALL    VPORT           ; configura p/ leitura
                                ;
                                ;configura usart:
        BSF     STATUS,RP0      ; -> seleciona banco1
        CLRF    SPBRG           ; configura taxa de tx em 1,25Mbps
        BSF     TXSTA,BRGH      ; configura baud rate em modo rapido
        BCF     TXSTA,SYNC      ; configura modo assincrono
        BSF     TXSTA,TXEN      ; habilita transmissao
        BCF     STATUS,RP0      ; -> seleciona banco0
        BCF     RCSTA,SREN      ; desabilita recepcao single
        BSF     RCSTA,CREN      ; habilita recepcao continua
        BSF     RCSTA,SPEN      ; habilita porta serial (pinos RC6 e RC7)
        BCF     TX_ON           ; habilita rx chip interface serial
                                ;
                                ;configura interrupcoes
        MOVLW   B'11010000'     ; mascara p/ configuracao
        MOVWF   INTCON          ; habilita interrupcao perif. e int.ext.
        BSF     STATUS,RP0      ; -> seleciona banco1
        MOVLW   B'00000111'     ; mascara p/ configuracao
        MOVWF   OPTION_REG      ; prescaler em 1:256 p/ tmr0
        BSF     PIE1,RCIE       ; habilita interrupcao de rx
        BCF     STATUS,RP0      ; -> seleciona banco0
```

```
;
;------------------------------ ROTINA PRINCIPAL ------------------------
;
MAIN:
       GOTO    MAIN           ;aguardando rx.
;
;------------------------------ SUB-ROTINAS CAMERA ----------------------
;
FPS:                          ;configura taxa de frames por segundo
       BCF     WEB            ;  zera WEB p/ nova escrita
                              ;  endereca registrador FRCTL (B'0100xxxx')
       BCF     A3             ;   A<3> em rb7 = 0
       BSF     A2             ;   A<2> em rb6 = 1
       BCF     A1             ;   A<1> em rb5 = 0
       BCF     A0             ;   A<0> em rb4 = 0
       MOVLW   B'00111111'    ;  seta SFR (B'xx111111') = 0,5 fps
       MOVWF   PORTD          ;  dado <- w
       NOP                    ;  delay p/ afirmar dados no barramento
       BSF     WEB            ;  efetua escrita (borda subida)
       RETURN
SFR:                          ;habilita single frame
       BCF     WEB            ;  zera WEB p/ nova escrita
                              ;  endereca registrador FCTL (B'0001xxxx')
       BCF     A3             ;   A<3> em rb7 = 0
       BCF     A2             ;   A<2> em rb6 = 0
       BCF     A1             ;   A<1> em rb5 = 0
       BSF     A0             ;   A<0> em rb4 = 1
       MOVLW   B'01000000'    ;  seta SFR (B'01000000')
       MOVWF   PORTD          ;  dado <- w
       NOP                    ;  delay p/ afirmar dados no barramento
       BSF     WEB            ;  efetua escrita (borda subida)
       RETURN
;
FSET:                         ;dispara single frame
                              ; habilita portD para saida
       BSF     STATUS,RP0     ;  -> seleciona banco1
       MOVLW   B'00000000'    ;  mascara p/ configuracao do portD
       MOVWF   TRISD          ;  portD <7:0>saidas
       BCF     STATUS,RP0     ;  -> seleciona banco0
                              ;
       BCF     WEB            ;  zera WEB p/ nova escrita
                              ;  endereca registrador FCTL (B'0001xxxx')
       BCF     A3             ;   A<3> em rb7 = 0
       BCF     A2             ;   A<2> em rb6 = 0
       BCF     A1             ;   A<1> em rb5 = 0
       BSF     A0             ;   A<0> em rb4 = 1
       MOVLW   B'11000000'    ;  seta FSET e mantem SFR (B'11000000')
       MOVWF   PORTD          ;  dado <- w
       NOP                    ;  delay p/ afirmar dados no barramento
       BSF     WEB            ;  efetua escrita (borda subida)
       RETURN                 ;encerra rotina
;
PCKS:                         ;habilita pclk apenas p/ pixel valido
       BCF     WEB            ;  zera WEB p/ nova escrita
                              ;  endereca registrador MCTL (B'0101xxxx')
       BCF     A3             ;   A<3> em rb7 = 0
       BSF     A2             ;   A<2> em rb6 = 1
       BCF     A1             ;   A<1> em rb5 = 0
       BSF     A0             ;   A<0> em rb4 = 1
       MOVLW   B'00000100'    ;  seta PCKS (B'00000100')
       MOVWF   PORTD          ;  dado <- w
       NOP                    ;  delay p/ afirmar dados no barramento
       BSF     WEB            ;  efetua escrita (borda subida)
       RETURN
;
HWCTL:                        ;configura limite horizontal
       BCF     WEB            ;  zera WEB p/ nova escrita
                              ;  endereca registrador HWCTL (B'0110xxxx')
       BCF     A3             ;   A<3> em rb7 = 0
       BSF     A2             ;   A<2> em rb6 = 1
       BSF     A1             ;   A<1> em rb5 = 1
       BCF     A0             ;   A<0> em rb4 = 0
       MOVLW   H'5A'          ;  seta bloco inicio(5) final(10) (5Ah)
       MOVWF   PORTD          ;  dado <- w
```

```
        NOP                     ;  delay p/ afirmar dados no barramento
        BSF     WEB             ;  efetua escrita (borda subida)
        RETURN
;
VWCTL:                          ;configura limite vertical
        BCF     WEB             ;  zera WEB p/ nova escrita
                                ;  endereca registrador VWCTL (B'0111xxxx')
        BCF     A3              ;   A<3> em rb7 = 0
        BSF     A2              ;   A<2> em rb6 = 1
        BSF     A1              ;   A<1> em rb5 = 1
        BSF     A0              ;   A<0> em rb4 = 1
        MOVLW   H'69'           ;  seta bloco inicio(6) final(9) (69h)
        MOVWF   PORTD           ;  dado <- w
        NOP                     ;  delay p/ afirmar dados no barramento
        BSF     WEB             ;  efetua escrita (borda subida)
        RETURN
;
VPORT:                          ;configuracao para leitura de dados da camera
                                ; habilita portD para entrada
        BSF     STATUS,RP0      ;  -> seleciona banco1
        MOVLW   B'11111111'     ;  mascara p/ configuracao do portD
        MOVWF   TRISD           ;  portD <7:0>entradas
        BCF     STATUS,RP0      ;  -> seleciona banco0
                                ; endereca registrador VPORT (B'10xxxxxx')
        BSF     A3              ;  A<3> em rb7 = 1
        BCF     A2              ;  A<2> em rb6 = 0
        BCF     OEB             ; habilita barramento dados
        RETURN                  ;encerra rotina
;
;------------------------------ SUB-ROTINAS USART -----------------------
;
TX_IDF:                         ;tx comando + endereco + frame completo
        BCF     RCSTA,CREN      ; desabilita recepcao
        BSF     TX_ON           ; habilita chip serial 485 a tx
                                ; tx comando:
        MOVLW   IDF             ;  w recebe comando a ser tx
        MOVWF   DADO            ;  salva dado a ser tx
        CALL    TX              ;  chama rotina p/ tx
                                ; tx endereco:
        MOVFW   PORTA           ;  w recebe endereco a ser tx
        MOVWF   DADO            ;  salva dado a ser tx
        CALL    TX              ;  chama rotina p/ tx
        RETURN                  ;
;
TX:                             ;tx dado
        BSF     STATUS,RP0      ; -> seleciona banco1
        BTFSS   TXSTA,TRMT      ; testa se buffer estah vazio
        GOTO    TX              ; loop enquanto buffer cheio
        BCF     STATUS,RP0      ; -> seleciona banco0
        MOVF    DADO,W          ; w recebe dado a ser tx
        MOVWF   TXREG           ; tx dado
DELAY_TX:                       ; aguarda buffer vazio p/ prosseguir
        BSF     STATUS,RP0      ; -> seleciona banco1
        BTFSS   TXSTA,TRMT      ;  testa se buffer estah vazio
        GOTO    DELAY_TX        ;  loop enquanto buffer cheio
        BCF     STATUS,RP0      ; -> seleciona banco0
        RETURN                  ;encerra rotina
;
;------------------------------ FIM DO PROGRAMA FONTE --------------------
        END                     ;obrigatorio
```