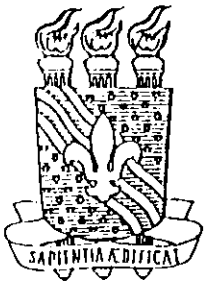


UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

RELATÓRIO DE ESTÁGIO

SHARP DO BRASIL S/A

Fernando Araújo Barros

Campina Grande, junho de 1998

Biblioteca Setorial do CDSA. Março de 2021.

Sumé - PB

ALUNO:

Fernando Araújo Barros

ORIENTADORES:

Prof. José Guttemberg de Assis

Sub. BRUNO BARBOSA ALBERT

Eng. Fernando Rangel de Sousa

AGRADECIMENTOS

Antes de tudo, agradeço a orientação do Deus Pai, que sempre me mostrou a que caminho seguir. Meus pais e minha futura esposa, pelo acompanhamento e compreensão das minhas atitudes. A todos professores que acreditaram no meu potencial, e me apoiaram durante minha passagem acadêmica.

Aos amigos e colegas paraibanos que hoje honram o nome desta instituição no Distrito Industrial de Manaus. E todos aqueles que de alguma forma se dispuseram a tornar a realização desse trabalho uma realidade.

INTRODUÇÃO

Este estágio tem como objetivo de preparar o aluno para seu ingresso no mercado de trabalho, assim como por em prática todo o conteúdo assimilado durante a graduação.

Na área da Eletrônica, a SHARP é uma grande empresa de desenvolvimento dos seus produtos, o qual se torna uma excelente oportunidade para os alunos da Engenharia Elétrica aprender e exercitar seus conhecimentos. Diversos alunos da UFPB passaram pela mesma experiência, trazendo excelentes resultados de seus trabalhos. Isto torna a SHARP uma das principais empresas que absorvem os alunos dessa instituição.

CRONOGRAMA

A Engenharia de Desenvolvimento de Produtos (EDP) da SHARP do Brasil, dentre outras atribuições, desenvolve projetos elétricos de receptores de TV. Visando preparar profissionais para atuar nesta área de conhecimento, anualmente a EDP oferece estágio para os alunos da Engenharia Elétrica.

Cronograma Proposto

Após o período de duração do estágio, o aluno deve estar apto a participar do desenvolvimento de novos produtos de vídeo. Para tal, o plano de estágio descrito posteriormente deve ser seguido.

O estágio foi previsto para o período de fevereiro a julho de 1998. As atividades a serem desenvolvidas devem obedecer o seguinte cronograma:

Etapa 1

- Integração com a empresa
- Familiarização com as atividades desenvolvidas na eng. de desenvolvimento
- Revisão dos conceitos básicos de televisão
- Elaborar diagrama de blocos detalhado de um TV moderno
- Apresentação semanal das atividades desenvolvidas
- Relatório parcial semanal
- Relatório parcial do período

Etapa 2

- Estudar o aparelho de TV modelo 29ST58 (circuitos)
- Elaborar diagrama de blocos do aparelho estudado.
- Compreender os circuitos e componentes integrantes de cada bloco do item anterior
- Apresentação semanal das atividades desenvolvidas
- Relatório parcial semanal
- Relatório parcial do período

Etapa 3

- Familiarização com os microcontroladores da família LC8641XX
- Estudar o software de controle do aparelho de TV modelo 29ST58
- Apresentação semanal das atividades desenvolvidas
- Relatório parcial semanal
- Relatório parcial do período

Etapa 4

- Elaborar especificação completa de software de um TV de nível intermediário.
- Elaborar os fluxogramas de todas as funções do software especificado
- Confeccionar bibliotecas de funções para os microcontroladores da família LC8641XX
- Apresentação semanal das atividades desenvolvidas
- Relatório parcial semanal
- Relatório parcial mensal
- Relatório final

	fevereiro	março	abril	maio	junho	julho
Etapa 1	XX					
Etapa 2		XX				
Etapa 3			XX			
Etapa 4				XX	XX	XX

Cronograma Executado

	fevereiro	março	abril	maio	junho	julho
Etapa 1	XX					
Etapa 2	XX	XX				
Etapa 3		XX				
Etapa 4		XX	XX	XX	-	-

HISTÓRICO

A SHARP do Brasil S/A - comumente denominada SDB, está situada no Distrito Industrial de Manaus - AM, onde possui três unidades: SBD I, SDB III e SBD IV. Nestas unidades, são produzidos aparelhos eletrônicos como TV'S, VÍDEOS, FORNOS MICROONDAS, AÚDIO, FILMADORAS e produtos para escritórios (calculadoras, fax, copiadoras, etc).

A empresa pertence ao Grupo Empresarial Machline que faz parceria com a SHARP CORPORATION do Japão, que faz uso da marca SHARP nos seus produtos, e desenvolve grande parte de seus produtos eletrônicos, como fornos microondas, *flybacks*, gabinetes de vídeo e áudio. Porém alguns produtos tem apresentam 100% de desenvolvimento aqui no Brasil, como os aparelhos de TV.

Esta é a principal função do Departamento de Engenharia e Desenvolvimento de Produtos / Imagem (EDP/I). Neste setor são desenvolvidos os aparelhos de TV, que torna a SHARP uma das líderes do mercado brasileiro.

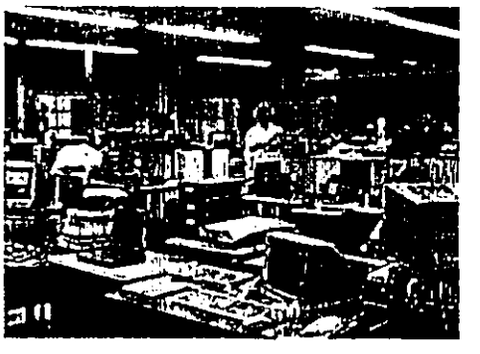


Figura 1: Departamento de Engenharia de Desenvolvimento de Produtos - Imagem

O desenvolvimento e a implantação de novos produtos na empresa consiste nas seguintes etapas:

- solicitação e avaliação de protótipos;
- processo 25 kits;
- linha piloto;
- início de produção.

Protótipo- Após a aprovação do Departamento de Marketing e do Departamento de Design de um determinado produto, a idéia é passada para o setor de Desenvolvimento Mecânico para a confecção do molde, com formas e design definidos. Em seguida a Engenharia de Produtos projeta o novo com seus devidos *features* (funções do produto) e requisita os componentes ao almoxarifado para a montagem de 5 protótipos. São realizados teste em campo (nos grandes centros urbanos), cujos defeitos são observados e corrigidos para a montagem dos 25 kits.

Processo 25 kits- O evento corresponde a montagem de 25 aparelhos com a responsabilidade do Departamento de Engenharia de Produtos, juntamente com o Departamento de Homologação de Produtos, que paralelamente executa teste nos circuitos e PCI (Placa de Circuito Impresso).

Linha Piloto- Após o fechamento dos 25 kits, dá-se início a linha piloto, que consiste na produção de 100 aparelhos. Nesta fase a responsabilidade pelo evento é da Engenharia Industrial. A Engenharia de Produtos neste etapa presta apenas suporte, e o fechamento do produto.

Início da Produção- O produto está pronto para ser produzido em larga escala e distribuído para todo país, tendo a garantia do Setor de Controle de Qualidade (CQ).

TV MODERNO

ELABORAÇÃO DO DIAGRAMA DE BLOCOS DETALHADO DE UM TV MODERNO.

De acordo com a Etapa 2 do cronograma proposto, foi elaborado o diagrama de blocos de um TV "moderno". Atualmente para os padrões da indústria brasileira, o TV, especificamente da SHARP, apresenta as seguintes características:

- *TV*-in-*TV*;
- Receptor estéreo;
- SAP;
- Controle remoto;
- Fone de ouvido sem fio;
- S-Vídeo (Y/C);
- Close-Caption;
- Vídeo-IN / Vídeo-OUT;
- CATV;
- OSD (One Screen Display);
- Microcontrolador.

Convém salientar que o diagrama elaborado é uma idéia preliminar do que se constitui um TV moderno, portanto não corresponde fielmente a realidade. Além disso não houve a preocupação em descrever cada circuito envolvido detalhadamente, mas apenas apresentá-los em blocos funcionais.

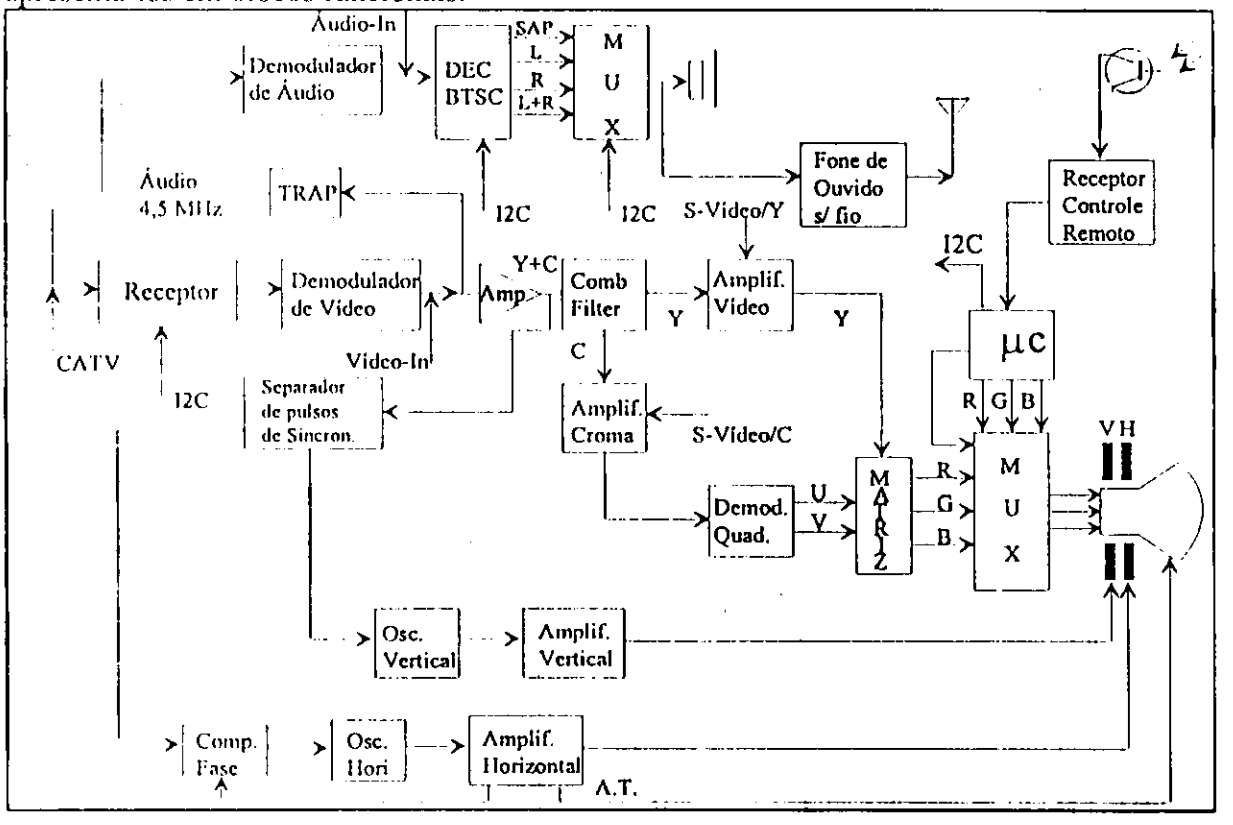


Figura 2:TV Moderno

Receptor

É responsável pela parte de recepção do sinal RF gerado pela emissora. O sinal é transladado para uma frequência intermediária (FI), através de um "batimento" com a frequência do oscilador local.

Demodulador de Vídeo

A frequência intermediária é composta de dois sinais modulados nas frequências de 41,25MHz (áudio) e 45,75MHz (vídeo). Em seguida há uma demodulação síncrona que aloca para a faixa de 0-6MHz (banda-base).

Trap

Circuito ressonante que captura o sinal de áudio, modulado em FM com portadora em 4.5MHz.

Demodulador de Áudio Estéreo

Faz a demodulação do áudio, podendo ser mono, estéreo ou SAP. Na saída do demodulador temos o sinal de áudio composto.

Decodificador Btsc

Seleciona o canal de áudio para amplificador de som. Este selecionamento é realizado pelo microcontrolador através do barramento I^2C.

Comb Filter

Separa o sinal de vídeo composto em luminância e crominância.

Amplificador de Croma

Amplifica sinal de croma na frequência da portadora de croma em 3,58 MHz. Neste estágio, podemos introduzir o sinal de croma do S-Vídeo.

Demodulador em Quadratura

Realiza a demodulação do sinal de croma e entrega os sinais U (R-Y) e V (B-Y) na entrada da matriz RGB.

Amplificador de Vídeo

Amplifica o sinal de luminância e entrega à matriz RGB. Na entrada do amplificador pode ser aplicado o sinal Y proveniente do S-Vídeo.

Separador de Pulsos de Sincronismo

Extrai e separa os pulsos de sincronismo do sinal de vídeo composto. A entrada de luminância do S-Vídeo também é aplicada nesse ponto.

Oscilador Vertical

Gera pulsos de sincronismo para o gerador de varredura vertical.

Oscilador Horizontal

Compara a fase do sinal de sincronismo do oscilador horizontal com os pulsos de sincronismo horizontal.

Amplificador Vertical e Horizontal

Amplifica os sinais de varredura vertical e horizontal, afim de elevar a corrente para excitar as bobinas de deflexão vertical e horizontal.

Receptor de Controle Remoto

Circuito dedicado a recepção de sinais provenientes do controle remoto que são entregues ao Microcontrolador.

Transmissor do Fone de Ouvido Sem Fio

Modula e transmite o sinal de áudio proveniente do decodificador BTSC.

Microcontrolador

Gerência todas as funções do televisor, usando para isso o protocolo de transmissão de sinais I^2C, e gera RGB próprio para a implementação do OSD (One Screen Display).

MODELO:29ST58

A seguir o diagrama de blocos de um televisor 29" modelo *MID-END*. Tal modelo apresenta as seguintes especificações:

- SAP;
- Estéreo;
- 3 entradas Áudio/Vídeo (AV);
- Entrada de Super-Vídeo (S-VHS);
- Fone de ouvido sem fio;
- 1 saída Áudio/Vídeo;

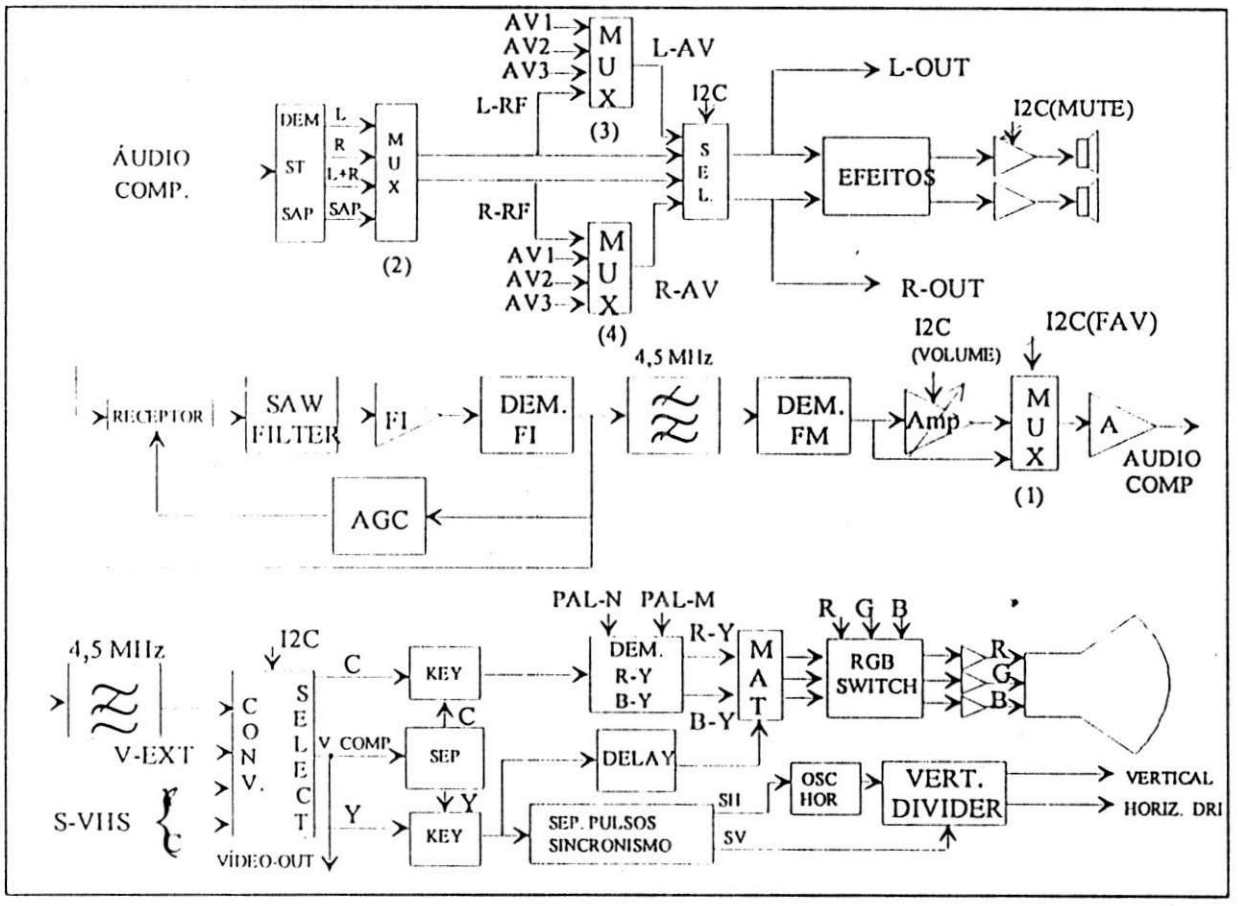


Figura 3: Diagrama de Blocos 29ST98

MUX (1)

Este mux seleciona entre o sinal na saída do amplificador com ganho variável ou o sinal proveniente do amplificador. Este procedimento existe porque, para algumas aplicações de processamento do sinal de áudio, necessita-se que o sinal a ser entregue apresente certos níveis de amplitude.

MUX (2)

Através deste dispositivo, seleciona-se entre sinal L, R, L+R ou sinal SAP.

MUX (3) e (4)

Responsável pela escolha da entrada do sinal de áudio, podendo ser AV1, AV2, AV3 ou áudio-RF, tanto para o canal esquerdo como o direito.

EFEITOS

São efeitos atribuídos a saída de áudio como volume, *bass*, *treble*, *loudness*, *subwoofer* e *surround*. Os sinais de R-OUT e L- OUT não estão sujeitos a esses efeitos.

CONVERSOR SELECT

Selecciona entrada do sinal de vídeo que pode ser sinal composto RF, entrada externa (AV) ou Super-Vídeo. A entrada V-EXT já foi previamente selecionada entre AV1, AV2 ou AV3. A entrada de Super-Vídeo é composta pelos sinais de luminância e crominância já separados. A seleção desses sinais é feita via protocolo I^2C.

RGB SWITCH

Bloco responsável pela a seleção do RGB proveniente do sinal de vídeo ou sinal gerado internamente pelo microcontrolador.

VERTICAL DIVIDER

Gera sinais dente-de-serra para varredura horizontal e vertical. Para perfeita sincronia utiliza sinais provenientes do oscilador horizontal e do separador de pulsos de sincronismo.

MICROCONTROLADOR LC8641XX *SANYO*

A série LC8641XX é um microcontrolador de 8-bits desenvolvido para aplicações de TV, com as seguintes características:

- decodificador de *closed caption* (*data slicer*);
- versões: 64k/ 56k/ 48k/ 40k/ 32k/ 24k/ 20k/ 26k/ 16k/ 12k-bytes de ROM;
- 384 bytes de RAM;
- 640 x 9-bytes CRT Display RAM (VRAM);
- 2 Timer/Contador 16-bits;
- 10 canais x PWM 4-bits;
- 4 canais x 4-bits Conversor A/D;
- interface serial síncrona 8-bits;
- 10 vetores de interrupção.

Uma placa emuladora ECB864100 possibilita uma simulação real do funcionamento do microcontrolador, para a realização dos testes, como mostra a Figura 4 abaixo.

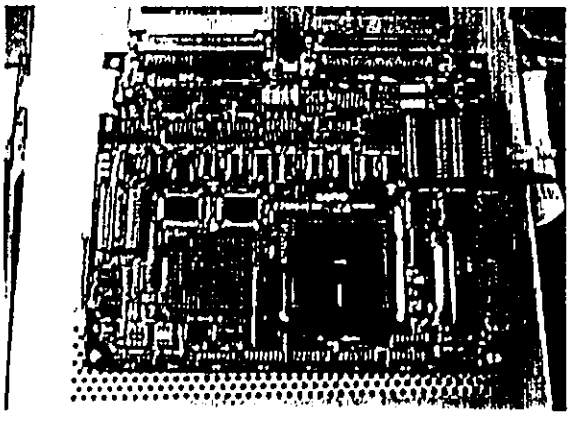


Figura 4: Placa emuladora LC86000

A seguir, na Figura 5 apresentamos o diagrama de blocos do microcontrolador série LC8641XX - *SANYO*.

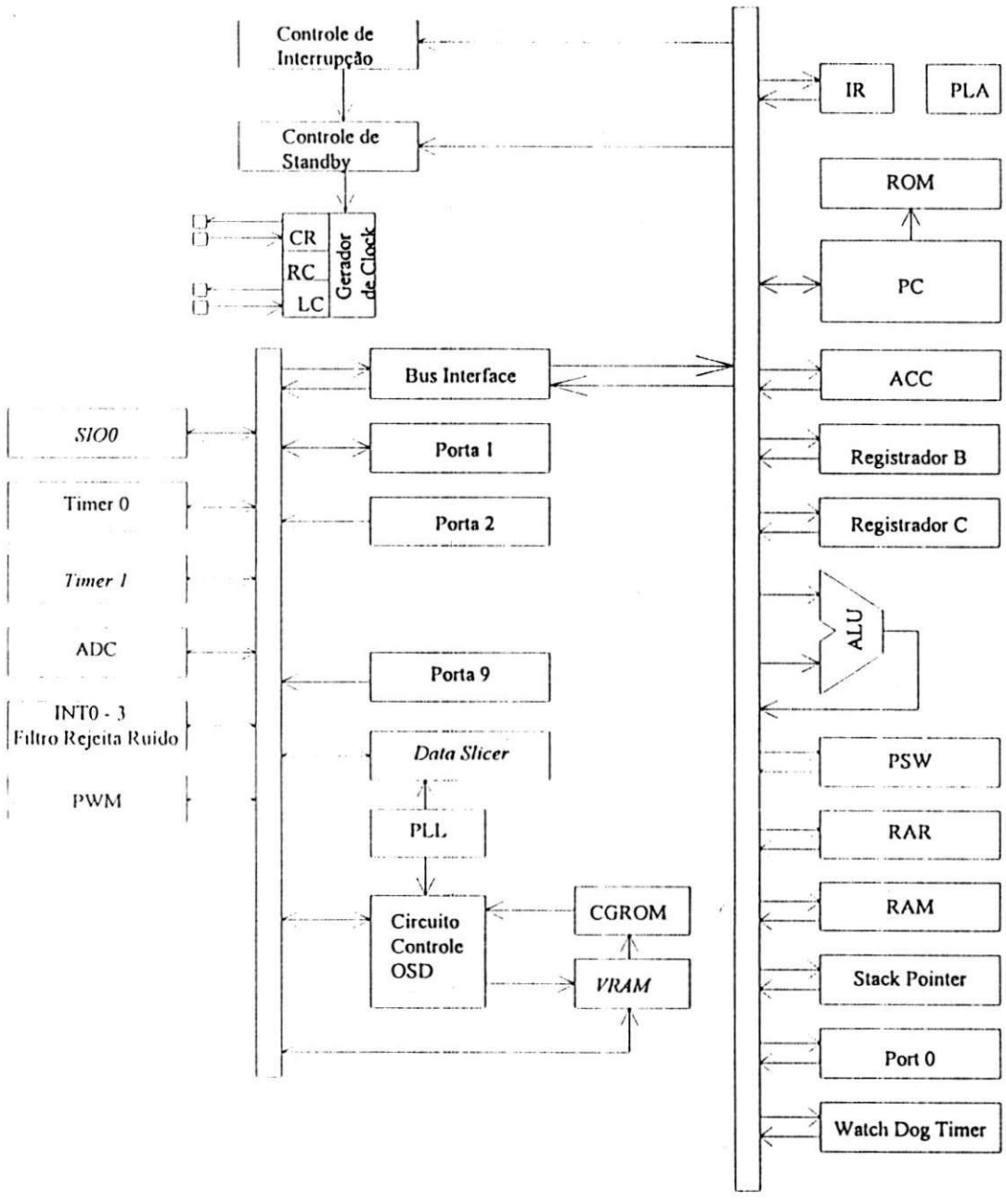


Figura 5: Diagrama de Blocos

SYSTEMA GERADOR DE CLOCK

LC8641XX apresenta dois circuitos osciladores para o sistema de gerador de *clock*: circuito oscilador *main clock*, e o circuito oscilador RC. Este sistema gerador é a base de tempo para execução das instruções. O sistema de clock pode ser seleccionado via *software*.

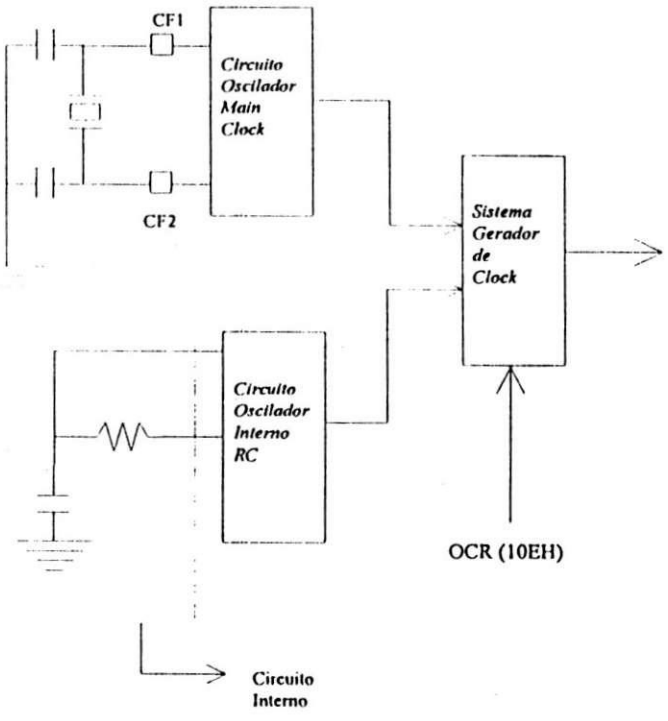


Figura 6: Sistema Gerador de *Clock*

- Circuito Oscilador *Main Clock*

Este circuito oscila através de uma ressonador cerâmico conectado aos terminais CF1 e CF2.

- Circuito oscilador interno RC

Este oscila aproximadamente em torno de 800 kHz através de um resistor R e um capacitor C.

- Seletor do sistema de *clock*

O sistema de *clock* fonte pode ser selecionado pelo *Oscillation Control Register* (OCR) bit-4 e bit-5.

Tempo do Ciclo de Instruções / Ciclo de Bus

Cada acesso a memória ROM é constituído de 2 ciclos de instruções. O ciclo de *bus* indica a velocidade de leitura da ROM.

Ciclo de Bus	Ciclo de Instrução	Sistema de Oscilação do Clock	Frequência de Oscilação	Tensão
0.5 μs	1.0 μs	Cerâmico (CR)	12 MHz	4.5 - 6.0 V
7.5 μs	15.0 μs	RC Interno	800 KHz	2.5 -6.0 V

MAPA DE MEMÓRIA

O Microcontrolador da série LC8641XX tem uma memória de programa (ROM) avaliada em até 64k bytes e uma memória de dados (RAM) de 512 bytes.

A memória de dados está dividida em 256 bytes de memória para dados (RAM, 000-0FFH), e 256 bytes para *Special Function Register* (SFR, 100-1FFH). A RAM consiste de dois bancos. O acesso a cada banco é dado pelo *program status word* (PSW), bit1 (RAMBK0) do SFR. O banco 0 é também usado como área de pilha.

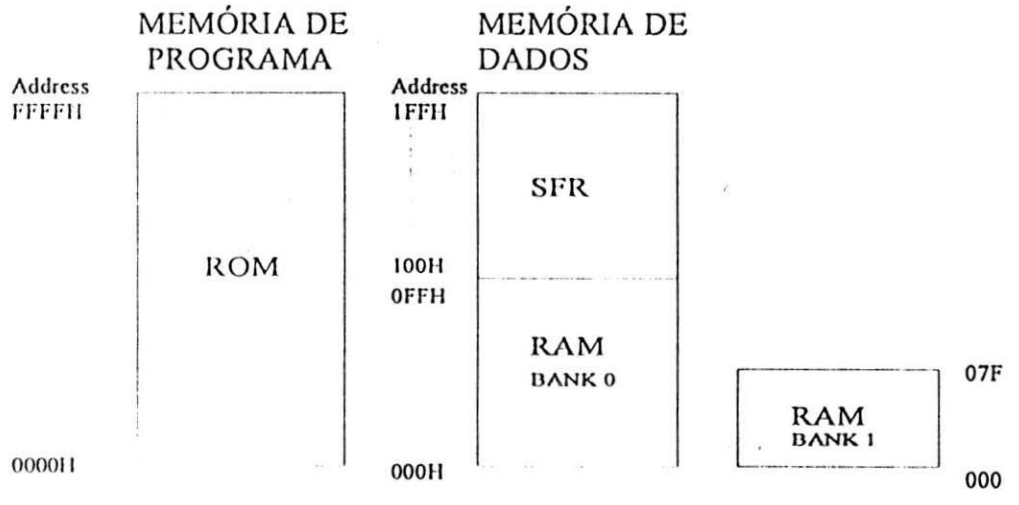


Figura 7:Mapa de Memória

Memória de Dados

Os primeiros 16 bytes (00h a 0FH) da memória RAM do LC8641XX são reservados para 4 bancos de registradores, com endereçamento indireto, @R0 e @R1 (para RAM), e @R2 e @R3 (para SFR). O tipo de endereçamento é realizado pela especificação do banco a ser utilizado, através do *Program Status Word* (PSW) bit 3 e 4 (IRBK0). Entretanto essa área também pode ser utilizada como área de RAM (normalmente).

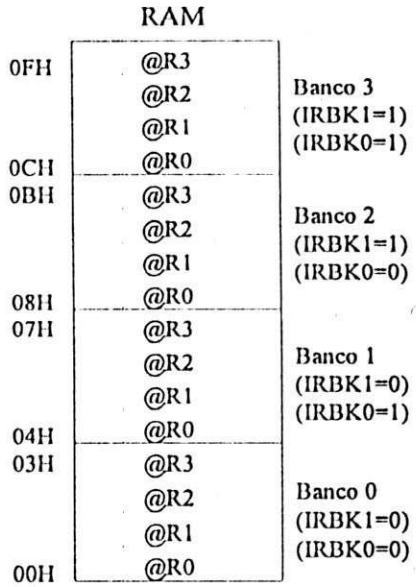


Figura 8: Registradores de Endereçamento Indireto

A tabela a seguir indica uma lista da Memória de dados, com suas respectivas variáveis de registros.

Símbolo	End.	R/W	Registrador
RAM (BANCO 0)	000H | 0FFH	R/W	Memória de Dados
RAM (BANCO 1)	000H | 07FH	R/W	Memória de Dados
ACC	100H	R/W	Acumulador
PSW	101H	R/W	*Program Status Word*
B	102H	R/W	Registrador B
C	103H	R/W	Registrador C
TRL	104H	R/W	*Table Reference Register Lower Byte*
TRH	105H	R/W	*Table Reference Register Upper Byte*
SP	106H	R/W	*Satck Pointer*
PCON	107H	R/W	*Power Control*
IE	108H	R/W	*Master Interrupt Enable Control*
IP	109H	R/W	*Interrupt Priority Enable Control*
OCR	10EH	R/W	*Oscillation Control*
WDT	10FH	R/W	*Watchdog Timer Control*
T0CNT	110H	R/W	*Timer 0 Control*
T0PRR	111H	R/W	*Timer 0 Prescaler Data*
T0L	112H	R	*Timer 0 Lower*
T0LR	113H	R/W	*Timer 0 Lower Reload Data*
T0H	114H	R	*Timer 0 Upper*
T0HR	115H	R/W	*Timer 0 Upper Reload Data*
T1CNT	118H	R/W	*Timer 1 Control*
T1LC	11AH	R/W	*Timer 1 Lower Comparison Data*
T1L	11BH	R	*Timer 1 Lower*
T1LR	11BH[1]	W	*Timer 1 Lower Reload Data*
T1HC	11CH	R/W	*Timer 1 Upper Comparison Data*
T1H	11DH	R	*Timer 1 Upper*
T1HR	11DH[2]	W	*Timer 1 Upper Reload Data*
OSDCR1	120H	R/W	*Caption Control 1*
OSDCR2	121H	R/W	*Caption Control 2*
ROWR	122H	R/W	*Display Row Address*
CLMR	123H	R/W	*Display Column Address*
VRAM	124H	R/W	*Video Ram*
HDSPR	125H	R/W	*Horizontal Display Start Position Control*
LNCR	126H	R/W	*Line Control*

[1] Endereço repetido, propositalmente
[2] Endereço repetido, propositalmente

Símbolo	End.	R/W	Registrador
LCCTR	127H	R	*Line Counter*
PPCR	128H	R/W	*Port Polarity Control*
FECR1	12BH	R/W	*Data Slicer Control 1*
FECR2	12CH	R/W	*Data Slicer Control 2*
VRCR	12DH	R/W	*Valid Reception Check*
CPDL	12EH	R/W	*Caption Data Lower*
CPDH	12FH	R/W	*Caption Data Upper*
SCON0	130H	R/W	*SIO0 Control*
SBUF0	131H	R/W	*SIO0 Buffer*
SBR	132H	R/W	*SIO Baud Generator*
P0	140H	R/W	*Port 0 Latch*
P0DDR	141H	R/W	*Port 0 Data Direction*
P1	144H	R/W	*Port 1 Latch*
P1DDR	145H	W	*Port 1 Data Direction*
P1FCR	146H	W	*Port 1 Function Control*
P7	15CH	R	*Port 7*
I01CR	15DH	R/W	*External Interrupt 0,1 Control*
I23CR	15EH	R/W	*External Interrupt 2,3 Control*
ISL	15FH	R/W	*Input Signal Selection*
P9	164H	R	*Port 9*
AD4CR	165H	R/W	*4-bit AD Converter Control*
PWM0	168H	R/W	*PWM 0 Comparison Data*
PWM1	169H	R/W	*PWM 1 Comparison Data*
PWM2	16AH	R/W	*PWM 2 Comparison Data*
PWM3	16BH	R/W	*PWM 3 Comparison Data*
PWM4	16CH	R/W	*PWM 4 Comparison Data*
PWM5	16DH	R/W	*PWM 5 Comparison Data*
PWM6	16EH	R/W	*PWM 6 Comparison Data*
PWM7	16FH	R/W	*PWM 7 Comparison Data*
PWM8	170H	R/W	*PWM 8 Comparison Data*
PWM9	171H	R/W	*PWM 9 Comparison Data*

Program Status Word

O PSW é um *flag* que indica o *status* do resultado de um calculo efetuado pelos registradores especifica o banco selecionado para a memória de dados RAM. Seu endereço na memória de dados é 101H e cada bit é inicializado com "0" no *reset*. Nesse registrador encontramos os seguintes *flag's*:

- *Carry Flag*;
- *Auxiliary Carry Flag*;
- *Indirect Address Register Bank Flag* 1;
- *Indirect Address Register Bank Flag* 2;
- *Overflow Flag*;
- *Data Memory Bank Flag*;

- *Parity Flag*;

Símbolo	End.	R/W	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PSW	101H	R/W	CY	AC	-	IRBK1	IRBK0	CV	RAMBK0	P
No reset			0	0	0	0	0	0	0	0

TIMER/CONTADOR

O LC8641XX é dotado de dois *timers*/contador internos.

Timer/Counter 0 (16-bits)

Apresenta as seguintes funções:

- Modo 0: 2 *reload timers*[3] independentes de 8-bits, manipulado por T0H e T0L, são operados por *prescaler clock*[4] de 10-bits;
- Modo 1: 2 *reload timer* 8-bits + *reload counter* 8-bits. T0H opera como um *reload timer* de 8-bits na mesma frequência do *prescaler clock*. T0L detecta e conta o sinal externo do terminais P72/INT2/T0IN e P73/INT3/T0IN;
- Modo 2: *reload timer de 16-bits*. T0H + T0L é operado pelo *prescaler clock* 10-bits;
- Modo 3: *reload counter* 16-bits. Este usa o *overflow* do registro T0L com o T0H *clock*, e opera como um contador de 16-bits. O T0L conta um sinal externo vindo dos terminais P72/INT2/T0IN e P73/INT3/T0IN;

Timer1

Apresenta as seguintes funções:

- Modo 0: 2 *reload timers* 8-bits;
- Modo 1: *reload timer* 8-bits + PWM 8 bits;
- Modo 2: *reload timers* 16-bits;
- Modo 3: PWM bit variável (9-16 bits).

INTERRUPÇÕES

Uma interrupção é uma função que suspende temporariamente o programa, o qual está em corrente execução pelo microcontrolador, e executa um outro programa que requer urgência. A série LC8641XX está implementada com circuitos que geram 12 tipos de requisições de interrupções, apresentados na tabela a seguir.

[3] Temporizador recarregável.
[4] clock programável

Prioridade	Tipo de Interrupção	Int./Ext.	Vetor de Endereço	Requisição da Interrupção
1	Interrupção Externa INT0	EXT	0003H	Detecta eventos do pino P0/INT0
2	Interrupção Externa INT1	EXT	000BH	Detecta eventos do pino P71/INT1
3	• Interrupção Externa INT2 • *Timer/Counter* T0L (*lower byte*)	• EXT • INT	0013H	• Detecta eventos do pino P72/INT2/T0IN • *Timer/Counter* T0L *overflow*
4	Interrupção Externa INT3	EXT	001BH	Detecta eventos do pino P73/INT3/T0IN
5	*Timer/Counter* T0H (*lower byte*)	IN	0023H	*Timer/Counter* T0H *overflow*
6	*Timer* 1	IN	002BH	• *Timer* T1L *overflow* • *Timer* T1H *overflow*
7	SIC0	IN	0033H	Detecta o fim de SIO0
8	*Data Slicer*	IN	003BH	O final da captura dos dados
9	VS\	EXT	0043H	Detecta o evento do pino VS\
10	Porta 0	EXT	004BH	Detecta nível "0" na Porta 0

A prioridades poder ser modificadas com a manipulação dos registradores IE (108H) e IP (109H).

OSD (On Screen Display)

O microcontrolador LC8641XX incorpora funções de OSD (*On Screen Display*) que possibilita a impressão de caracteres na tela do TV incluindo funções para *close caption.*

CARACTERÍSTICAS

Pode-se enumerar abaixo algumas das principais características incorporadas ao microcontrolador:

- Tela para *display* : 34 colunas x 16 linhas;
- *Display* para RAM : 640 x 16 linhas x 9 bits;
- Atributos de caracteres
    1. Cor de caracter;
    2. Cor de fundo do caracter;
    3. Sombreamento;
    4. Cor de tela cheia;
    5. Sublinhado;
    6. Itálico;
- Escolha da posição vertical a serem inseridos caracteres na tela do TV;
- Escolha da posição horizontal a serem inseridos caracteres na tela do TV;
- Especificação do modo dos caracteres a serem exibidos por linha (*close caption*, modo texto, modo *OSD*);
- Oito tipos de tamanho do caractere : Horizontal x Vertical = 1x1, 1x2, 2x2, 2x4, 1.5x1, 1.5x2, 3x2, 3x4;

Para a geração de caracteres na tela da TV, é necessário a configuração de alguns Registradores de Funções Especiais (SFRs), que controlam o OSD. Nesses registradores são guardados dados de 96 palavras (16 linhas x 6 colunas) na VRAM e nos SFR's, relacionados abaixo com algumas de suas características:

- OSDCR1 (120H)
  Cor de tela cheia (*full screen*);
  Tela cheia ON/OFF
  Controle de *flash*
  Operação OSD ON/OFF;
- OSDCR2 (121H)
  Habilita controle de interrupção do sincronismo vertical (VS);
  Bit-8 da VRAM da dados para leitura;
  Bit-8 da VRAM de dados para escrita;
- ROWR (122H);
  Acesso da linha na VRAM;
- CLMR (123H);
  Acesso da coluna na VRAM;
- VRAM (124H);
  Armazena os dados que serão escritos na tela;
- HDSPR (125H);
  Especifica a posição horizontal de todas as linhas na tela;

- PPCR (128);

  Especifica a polaridade de cada sinal (R, G, B, I, BL, VS e HS);

O OSD pode ser controlado em cada linha da VRAM, e cada linha da VRAM corresponde a uma linha do *display*. O circuito OSD converte os dados da VRAM em R, G, B, I e BL de acordo com os atributos dos dados de controle da VRAM.

VRAM

A VRAM (*Display RAM*) tem capacidade de 640 x 9 bits que é constituída de 16 linhas x 40 colunas x 9 bits. Os seis primeiros bytes de cada linha controla as características da respectiva linha da VRAM (dado de atributo). Estas seis primeiras palavras não são usadas como dados de *display*. Restam então 34 palavras que são usadas como dados de *display*. Nestas região de *display* os dados de caractere e dados de atributo podem ser colocados em posições arbitrárias. Quando o bit-8 da VRAM é "0", isto significa que é um dado de caractere. E quando é "1", significa que é um atributo.

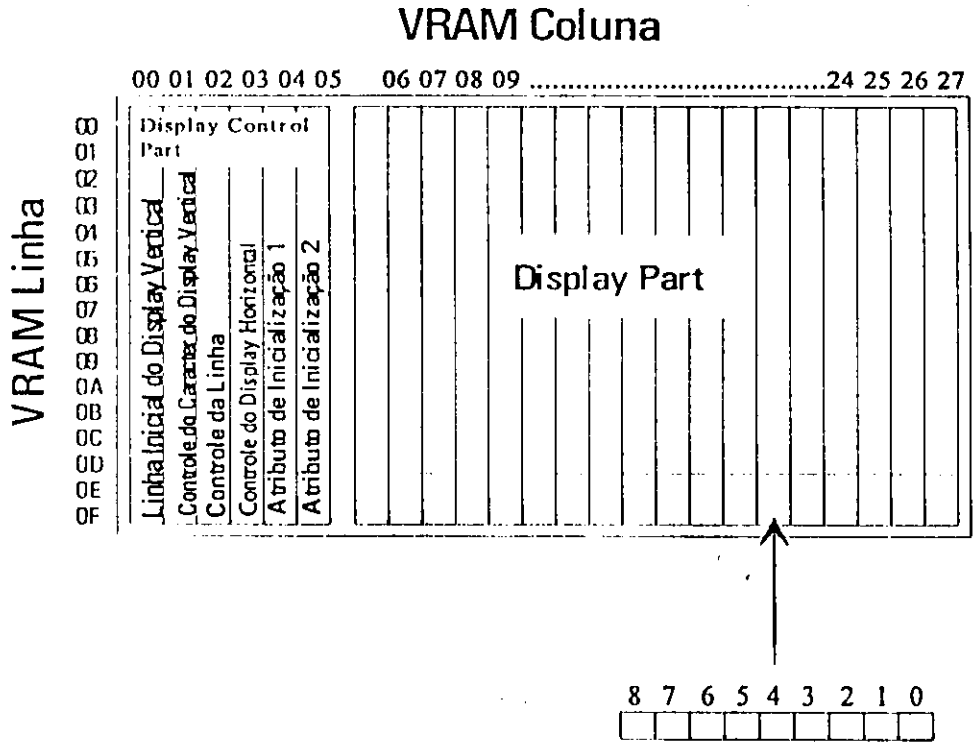


Figura 9: VRAM Layout

CONTROLE DO DISPLAY

Apresentamos abaixo, de forma resumida, as funções dos registadores da VRAM

- Vertical *Display Starting Line Number* (VDSLN) - Configura o número da linha de traço horizontal da tela onde será apresentada cada linha da VRAM. Em outras palavras, o conteúdo de VDSLN controla a posição vertical de cada linha de dados da VRAM que será mostrada na tela.

- Vertical *Character Display Control* (VCDC) - Este registrador controla o início ou final do efeito *Shutter* de cada caractere
- *Row Control* (RCON) - Controla o tamanho do caracter, o modo do *display* (OSD 1, OSD 2, *Caption*, Texto), intensidade do *background*, cor do *background*.
- Horizontal display Control - Controla a distância entre caracteres numa mesma linha.
- Atributos de Inicialização - Inicializa os atributos de cada linha. Entre outras funções, eles realizam: controle de *flash*, itálico, sublinhado, cor do caractere, intensidade da cor de fundo do caractere.

GERADOR DE CARACTERE (CGROM)

Podem ser registrados 256 caracteres na CGROM. Há um número de 128 caracteres que são compostas de tamanho 9 x 9 dots. Existem alguns endereços reservados como por exemplo, 020H que é para espaço sólido. É necessário colocar os caracteres na CGROM para serem utilizados no *display*.

PROTOCOLO I^2C

Protocolo I2C suporta tanto a tecnologia NMOS como CMOS. Dois condutores denominados SDA (linha de dados) e SCL (linha de *clock*) transmitem informação entre dispositivos conectados ao barramento. Cada dispositivo é reconhecido por um único endereço e podem operar tanto com transmissor ou receptor. O dispositivo que inicia a transferência de dados e gera o sinal de clock é denominado *master,* enquanto o dispositivo que recebe dados do barramento é chamado de *slave.*

Características Gerais

As linhas SDA e SCL são bidirecionais, conectadas a uma fonte de tensão positiva via resistores de pull-up. Quando o barramento está ocioso ambas as linha estão em nível alto de tensão. Os estágios de saída dos dispositivos devem ser coletor aberto (ou dreno aberto) para prover a função de *wired-AND.* Os dados no barramento I2C podem ser transferidos a uma taxa de até 100 kbits/s no modo padrão. O número de interfaces conectadas ao barramento é limitada pela capacitância máxima permitida ao barramento que é de 400 pF.

Os dados na linha SDA devem estar estável durante o nível alto do clock . Os níveis alto e baixo da linha SDA só podem ser modificados quando houver nível baixo da linha SCL.

Inicio e Final da Transmissão

Protocolo I2C define situações únicas para definir condições de início e final da transmissão. Uma transição de nível alto para baixo, enquanto o SCL está em nível alto, significa início da transmissão (START). Uma transição de nível baixo para alto, enquanto SCL está em nível alto, significa final da transmissão (STOP).

As condições de START e STOP são sempre geradas pelo *master.* O barramento é considerado ocupado depois da condição de START e ocioso depois do STOP.

Figura 10:Bits de *Start* e *Stop*

Transferência de Dados

O número de bytes que pode ser transmitido por transferência é ilimitado. Todo byte transferido tem que ser acompanhado por um bit de reconhecimento (ACKNOWLEDGE BIT). Os dados são transferidos primeiramente com MSB

Se o receptor não puder receber algum outro byte completo de dados, até que ele tenha executado qualquer outra função, ele pode segurar a linha de clock em nível baixo para forçar a transmissão para um estado de espera. A transferência de dados então continua e quando o receptor estiver pronto para algum outro byte de dados ele libera a linha SCL.

Acknowledge (ACK)

Transferência de dados com ACK é obrigatório. O pulso de clock do ACK é gerado pelo master. O transmissor libera a linha de dados (HIGH) durante o pulso de clock do ACK.

O receptor deve forçar a linha SDA para nível baixo de maneira tal que ela esteja estável durante o nível alto do pulso de clock.

Quando o escravo-receptor não reconhece o endereço (por exemplo, está executando alguma função em tempo real) a linha de dados deve ser deixada em nível alto para que o mestre possa gerar uma condição de STOP.

Se um escravo-receptor reconhece o endereço, mas se algum tempo depois na transferência não puder receber mais dados, o master deve novamente abortar a transferência através de uma condição de STOP.

Um master-receptor deve sinalizar o fim de uma transferência gerando um ACK no último byte transferido pelo escravo. O escravo-transmissor deve liberar a linha de dados para permitir o master gerar um STOP ou uma repetida condição de START.

Abaixo apresentamos o diagrama de tempos e modos do protocolo I2C.

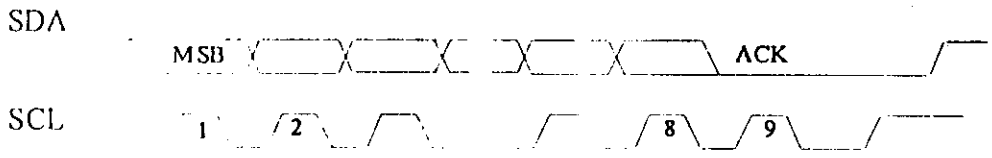


Figura 11: Protocolo I2C

Operação do Dispositivo:

Foi utilizada a memória EEPROM ST24C01 de 1 kbits que é compatível com o padrão I2C. A memória possui um endereço único de 4 bits que é usado juntamente com 3 *chip enable* que são individualmente selecionados pelo usuário. Assim, até 8 memórias de 1 kbits podem ser conectadas em um barramento I2C.

Operação de Escrita

Há três tipos de escrita suportados pelo ST24C01:

- **Byte Write:** Neste modo, o mestre manda um byte o qual é reconhecido pela memória. O mestre então termina a transferência com a geração da condição de STOP;
- **Multibyte Write:** Este modo pode ser inicializado em qualquer endereço na memória. O mestre envia de 1um até 4 bytes de dados que são reconhecidos independentemente pela memória. Escrever mais de 4 bytes neste modo pode modificar os bytes em linhas adjacentes (uma linha tem oito bytes) Entretanto se a escrita começar no primeiro endereço destes oito bytes que é o início da linha, pode-se escrever até oito dados neste modo;
- **Page Write:** Este modo permite a escrita de até 8 bytes num único ciclo de escrita. O mestre envia de um até 8 bytes de dados e, para cada byte escrito, a memória envia o sinal de acknowledge.

Operação de Leitura

Há três tipos de operação de leitura:

- **Leitura de Endereço Atual:** A memória tem internamente um contador de endereço. Cada vez que um byte é lido, um contador é incrementado. Neste modo, depois da condição de START, o mestre envia o endereço da memória com o R/W setado para 1. A memória o reconhece e envia para o barramento o conteúdo da memória endereçada no contador interno que, em seguida, é incrementado. O master não manda o ACK, mas termina a transferência gerando a condição de STOP;
- **Leitura de Endereço Aleatório:** Um escrita irrelevante é utilizada para carregar o endereço desejado no contador de endereço. Isto é seguido por outro START seguido do endereço do dispositivo e com o bit R/W setado para 1 realizado pelo mestre.;
- **Leitura Seqüencial:** Este modo permite ser iniciado com qualquer dos modos anteriores. Entretanto, neste caso, o mestre manda o sinal de ACK e o dispositivo continua a envia os demais bytes. Para terminar a transmissão o mestre manda o NOT ACK e gera a condição de STOP.

CLOSED CAPTION (FCC 47 CFR 15.119)

O Microcontrolador série LC8641XX permite a extração dos dados do sinal *closed caption*. O sinal consiste de dados codificados na linha 21 do retraço vertical do sinal de vídeo composto, podendo ser utilizado para transmissão de texto ou como Serviços de Dados Extendidos (*Extended Data Services*).

Em alguns países esse serviço é regulamentado por lei e torna obrigatório para todos os aparelhos receptores. Nos Estados Unidos este padrão é determinado pela *Federal Communications Commission.*

O sinal de dados na linha 21 consiste de dados independentes no campo 1 ou campo 2. Cada canal de dados pode conter tipos específicos de pacotes de dados como mostrados a seguir.

CAMPO 1	CAMPO 2
CC1 (Serviço de Captação Síncrono Primário)	CC3 (Serviço de Captação Síncrono Secundário)
CC2 (Captação Especial Não-Síncrono)	CC4 (Captação Especial Não-Síncrono)
T1 (Serviço de Texto 1)	T3 (Serviço de Texto 3)
T2 (Serviço de Texto 2)	T4 (Serviço de Texto 4)
	EDS (Serviço de Dados Extendidos)

O Serviço de Captação Síncrono Primário (CC1) é a captação de dados referente a língua primária que devem estar em sincronia com o áudio, do respectivo quadro da imagem. O Serviço de Captação Síncrono Secundário (CC3) é um canal de captação de dados alternativo, usualmente utilizado para um segundo canal de áudio

O canal especial Não-Síncrono (CC2, CC4), transmite dados que incrementam a informação do programa corrente, e não necessita estar em sincronismo.

Forma de Onda

Abaixo na Figura 12 apresentamos a forma de onda do sinal *closed caption*. O *Clock Run-In* é uma senóide simétrica com amplitudes máxima e mínima iguais aos níveis lógicos "0" e "1",respectivamente, que iniciam os dados codificados. Ele também está em fase com todos os níveis de transição dos *Start* e *Data Bits*. Três *Start Bit* seguem as mesmas especificações como também os *Data Bits*, mas sempre definidos pelos níveis "0", "0" e "1". Na podemos observar o sinal real proveniente do gerado de sinais padrões para TV.

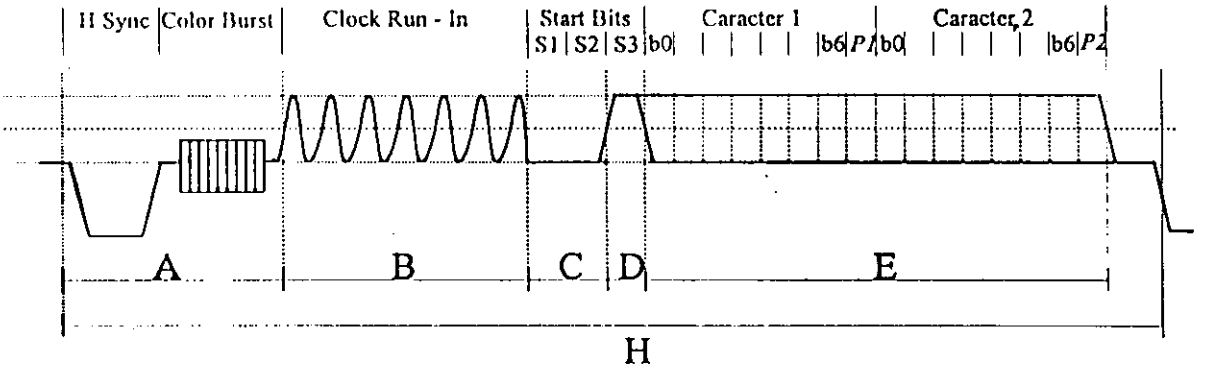


Figura 12: Sinal *Closed Caption* (linha 21)

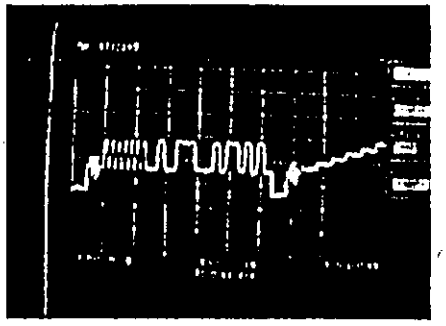


Figura 13: Sinal *Closed Caption* no osciloscópio.

A tabela a seguir contem informações para codificadores e decodificadores de *closed caption*. As especificações para as duas seções podem ser diferentes (tolerância ±5%), pois o sinal está sujeito a distorções introduzidas no canal durante a transmissão.

Intervalo	Descrição	Codificador	Decodificador
A	*H-Sync* para *Clock-Run-In*	10.500 µs	10.500 µs
B	*Clock-Run-In*	12.910 µs	12.910 µs
C	*Clock-Run-In* para 3° *Start Bit*	3.972 µs	3.972 µs
D	*Data Bit*	1.986 µs	1.986 µs
E	*Data Characteres*	31.778 µs	31.778 µs
H	Linha horizontal	63.556 µs	63.556 µs

Norma para Decodificadores de Closed Caption *em Receptores de TV*

Abaixo apresentamos as regras para **codificação** e **decodificação** do *closed caption*, segundo a Federal Communication Commission (FCC 47 CFR Part 15.119) que, futuramente, deverá ser adotado pelo Brasil.

Formato de Transmissão - A informação de *closed-caption* é transmitida na linha 21 do campo 1 do intervalo de *blanking* vertical do sinal de televisão.

Modo de Operação - O receptor de televisão deverá suportar modos de operação para TV e *closed-caption*. Um terceiro modo de operação, texto, poderá ser incluído opcionalmente. Os modos texto e *caption* contém dados em qualquer dos dois canais de operação (C1 e C2). O receptor deve decodificar tanto C1 e C2 e deverá apresentar o *captioning* em qualquer canal que o usuário selecionar. O modo *caption* e texto definirão um ou mais áreas na tela onde o *caption* ou o texto serão apresentados.

Formato da Tela - A área de display para *captioning* e texto deverá estar contida aproximadamente na área definida como *safe area caption* (área reservada). A área de *display* deve ser dividida em no máximo 15 linhas de caracteres de igual espaçamento e 32 colunas de igual largura. Verticalmente a área de *display* deve se iniciar na linha 43 e ter 195 linhas de altura, terminando na linha 237 (tela entrelaçada). Todos os textos e *captionings* devem estar dentro destes limites e devem estar claramente separadas da imagem de vídeo, como mostra a Figura 14.

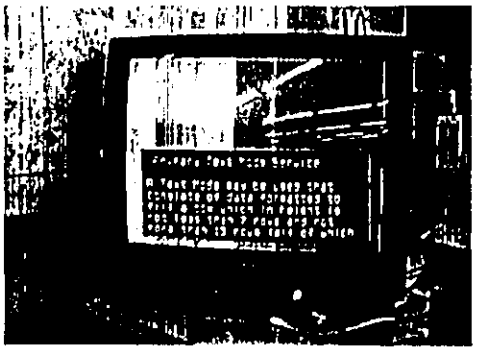


Figura 14: Apresentação do Modo Texto

Modo Caption

Neste modo, o texto deve aparecer em até 4 linhas simultaneamente em qualquer área da tela dentro de uma área definida de *display*. Em adição, um espaço sólido igual a uma coluna de largura deve ser colocada antes do primeiro e após o último caractere da linha com a finalidade de melhorar a legibilidade. A área de *caption* deve ser transparente se:

Nenhum caractere de espaço padrão ou outro caractere foi endereçado e não foi necessário nenhum espaço sólido.

Um acompanhamento de espaço sólido é usado e um caractere especial "espaço transparente" foi endereçado que não precede ou segue imediatamente um caractere imprimível.

Formato de apresentação - Com intuito de analisar os caracteres de apresentação, é conveniente pensar em termos de um cursor não visível que marca na tela a posição em que o próximo evento num determinado modo e canal ocorrerá. O receptor relembra a posição do

cursor para cada modo, mesmo quando os dados são recebidos para diferentes endereços em diferentes modos e canais.

Endereçamento na tela - Dois tipos de códigos de controle são usados para mover o cursor para um locação específica na tela. No modo *caption*, estes códigos de endereçamento podem afetar tanto posição da linha quanto da coluna. No modo texto, os códigos afetam apenas posição da coluna. Em ambos os modos, os códigos de endereçamento são opcionais. Posição específicas são definidas para cada modo e estilo quando nenhum código de endereçamento é recebido.

O primeiro tipo de código de endereçamento é *Preamble Address Code* (PAC). Ele assinala o número da linha e uma de oito tipos de indentações. Cada indentação move o cursor 4 colunas para direita (começando pela margem esquerda). Neste modo, uma indentação de 0 coloca o cursor na coluna 1, uma indentação de 4 coloca o cursor na coluna 5 e assim por diante. A indentação PAC é não-destrutiva, ou seja, ele não afeta os caracteres a esquerda da nova posição do cursor. O PAC também determina os atributos dos caracteres seguidos do código.

O segundo tipo de código de endereçamento é o *Tab Offset* que é um de três *Miscelaneous Control Codes*. *Tab Offset* move o cursor uma, duas ou três colunas para direita. Os caracteres, então se moverão para direita sem serem afetados.

Existem três tipos de estilos de apresentação de texto no Modo C*aption*:

- *roll-up*
- *pop-on*;
- *paint-on*.

Roll-up

Este estilo é iniciado pela recepção de um de três *Miscelaneous Control Codes* que determinam o número máximo de linhas mostradas simultaneamente que podem ser 2, 3 ou 4 linhas.

A linha mais abaixo do *display* é chamada da linha de base (*bottom row*)e o cursor sempre termina nesta linha. Linhas do texto rolam para cima continuamente para criar uma janela de 2 a 4 linhas.

O comando *roll-up* normalmente é seguido (não necessariamente imediatamente) por um PAC indicando a base da linha e a posição horizontal de indentação. Se nenhum PAC é recebido, a base da linha deve ser a linha 15 (por *default*) ou, se um *roll-up caption* é correntemente exibido, a próxima linha de base deverá ser exibida na mesma linha de base que foi recebida anteriormente e o cursor deve ficar na coluna 1. Se o PAC recebido contém uma linha de base diferente da do *caption* correntemente exibido, então a janela imediatamente se move intacta e sem apagamento para nova linha de base.

Cada vez que um *carriage return* é recebido, o texto da linha de topo da janela é apagado da memória e da janela.

Aumentando ou diminuindo o número de linhas de *roll-up* muda-se instantaneamente o tamanho da janela exibida atualmente.

Caracteres devem ser exibidos imediatamente ao serem recebidos pelo receptor. Uma vez o cursor ter atingido a coluna 32, todo caractere subseqüente anterior ao PAC, retorno de carro e *backspace* deve ser exibido na mesma coluna apagando o caractere que ocupava aquela posição.

O cursor move automaticamente uma coluna a direita depois de cada caractere ou *Mid Row Code* recebido. Um *backspace* move o cursor para direita apagando o caractere ou Mid Row Code ocupando aquela posição. Um *backspace* na coluna 1 é ignorado.

O comando *Delete to End of Row* apaga da memória qualquer caractere ou código de controle começando no cursor corrente e em todas as colunas a sua direita na mesma linha.

Se um espaço sólido é usado para legibilidade, ele deve aparecer quando o primeiro caracter imprimível (não um caractere transparente) ou *Mid-Row Code* (MRC) é recebido numa linha, não quando um PAC, se algum, é dado. Uma linha em que não exista o caractere imprimível um MRC na imprimirá uma espaço sólido, mesmo quando rolada entre duas linhas que imprimam um espaço sólido.

Se a recepção de dado de uma linha é interrompida por um dado de um outro canal ou pelo modo texto, a exibição do texto *caption* continuará da mesma posição do cursor se um comando roll-up for recebido e nenhum PAC é recebido para mover o cursor.

O *caption roll-up* permanecerá exibido até a recepção de uma das técnicas padrão de apagamento. Recepção de comando *Resume Caption Loading Command* (para *Pop-On)* ou um comando *Resume Direct Captioning* (para *paint-on*) não afeta a exibição do *Roll-up.*Recepção do comando *Roll-up Caption* causa apagamento da memória imprimível e não imprimível dos estilos *Pop-On* e *Paint-On.*

Pop-on

Este estilo é iniciado ao receber o comando *Resume Caption Loading.* Subseqüentemente os dados são carregados numa memória, mas sem exibir na tela, até que um comando de End of Caption seja recebido e a partir deste momento o conteúdo da memória passa a ser exibido na tela e vice-versa sem apagar automaticamente a memória. Um *End of Caption* força o receptor a entrar no estilo *pop-on* se nenhum comando *Resume Caption Loading* tenha sido recebido.

PAC pode ser usado para mover o cursor em torno da tela em ordem aleatória para colocar o caption nas linhas 1 a 15. O *carriage return* não tem efeito na localização do cursor durante carregamento do caption.

O cursor move automaticamente uma coluna para direita depois de cada caractere ou *Mid Row Code* recebido. A recepção de um *backspace* move o cursor uma coluna para esquerda, apagando um caractere ou MRC ocupando aquela posição. Quando o cursor

alcança a coluna 32 em qualquer linha, todos caracteres subsequentes recebidos antes do *backspace*, *end of caption* ou PAC substituem o caractere naquela posição.

O comando *Delete to End of Row* apaga da memória qualquer caractere ou código de controle começando na localização do cursor corrente e em todas colunas a sua direita na mesma linha.

Se a recepção de dados é interrompida durante o carregamento do *caption* por dados de outro canal de *caption* ou pelo modo texto, o carregamento do *caption* se iniciará na mesma posição do cursor corrente caso seja recebido um *Resume Caption Loading* e nenhum PAC tenha sido recebido para mover o cursor.

Caracteres mantém-se numa memória sem exibição na tela até que seja recebido um comando de *End of Caption*. O *caption* será apagado sem que seja exibido se for recebido um comando *Erase Non displayed Memory*, um comando *Roll-Up Caption*, ou se o usuário mudar o cana de recepção, canal de dados ou campo, ou se ainda houver perda de caracteres válidos.

Um *caption Pop-On*, uma vez exibido, é mantido na tela até que seja recebido um código de apagamento ou até que um comando de Roll-Up seja recebido.

Paint-On

É iniciado pela recepção do comando *Resume Direct Captioning*. Subseqüentemente os dados são apresentados na tela imediatamente sem a necessidade do comando *End of Caption*.

PAC é usado para mover o cursor em torno da tela em ordem aleatória nas linhas de 1 a 15. *Carriage returns* não tem qualquer efeito na localização do cursor durante o *captioning* direto. O cursor move-se automaticamente uma coluna a direita depois de cada caractere ou MRC. Um *backspace* move o cursor uma coluna para esquerda apagando o caractere naquela posição. Após o cursor alcançar a coluna 32 de cada linha, todo o caractere recebido antes do PAC ou *backspace* será exibido naquela coluna sobrescrevendo o caractere anterior.

O comando *Delete to End of Row* apaga da memória qualquer caractere ou código de controle começando na localização do cursor corrente e em todas colunas a sua direita na mesma linha. Se manter algum caractere não imprimível na linha depois do comando *Delete to End of Row*, o espaço sólido (se existir algum) para aquele elemento deve ser apagado.

Se a recepção de dados é interrompida durante o carregamento do *caption* por dados de outro canal de *caption* ou pelo modo texto, o carregamento do *caption* se iniciará na mesma posição do cursor corrente caso seja recebido um *Resume Caption Loading* e nenhum PAC tenha sido recebido para mover o cursor.

Caracteres mantém-se imprimidos até uma técnica padrão de apagamento seja aplicada ou um comando de *Roll-Up* seja recebido. Um *End of Caption* deixa um *caption* totalmente intacto na memória não imprimível

Tabela de Caracteres

Caracteres especiais - Estes requerem dois bytes para cada símbolo. Cada código hexadecimal deve ser precedido por um 11h para canal de dados 1 ou por um 19h para canal de dados 2.

Atributos de caracteres - Um caracter pode ser transmitido com qualquer dos 4 atributos: Cor, itálico, sublinhado e *flash*. Todos estes atributos são configurados pelos códigos de controle incluídos nos dados de recepção. Um atributo se manterá em efeito até o final da linha ou até a chegada de outro código de controle. Cada linha é iniciada por um código de controle que configura cor e sobrescrito. O *default* é caractere branco sem sobrescrito.

Todos MRC e o comando *Flash On* são atributos de espaço que aparecem na tela como um espaço padrão (20h). PAC não provoca espaço padrão.

O atributo de cores tem a mais alta prioridade e pode ser somente modificado pelo MRC de outra cor. Itálico tem a segunda maior prioridade. Se um caracter tanto com cor e itálico é desejado, o MRC itálico deve seguir o atributo de cor. Qualquer MRC desabilitará itálico. Se o último bit significativo de uma PAC ou de uma cor ou itálico MRC for 1, sublinhado estará habilitado.

O atributo de *Flash* é transmitido como *Miscellaneous Control Code*. O *flash On* não altera o status da cor, itálico ou subscrito. Entretanto, qualquer cor ou itálico MRC desabilitará o *flash*.

Por exemplo, se desejamos um caracter vermelho, itálico, sublinhado e *flash*, os atributos devem ser recebidos na seguinte ordem: um vermelho PAC ou MRC, um itálico MRC com bit de sublinhado e o comando de *Flash On*.

Exibição de Atributos

O atributo de sublinhado é exibido por uma linha abaixo do caractere da mesma cor do caractere. O atributo de flash é exibido de tal maneira que cause um apagamento de pelo menos uma vez por segundo. O suporte de cor para os atributos é opcional. Se não houver suporte para cores, todos os atributos de cores serão ignorados.

Códigos de controle - Existem três tipos diferentes caracteres de controle usados para para identificar o formato, localização, atributos e exibição de caracteres. *Preamble Address Code* (PAC), *Mid Row Code* (MRC) e *Miscellaneous Control Code* [5].

Cada código de controle consiste de um par de bytes que são sempre transmitidos juntos em uma única linha de campo (21) e que são normalmente transmitidos duas vezes para aumentar a confiabilidade de uma recepção correta. O primeiro byte dos códigos de controle é um caractere não imprimível na faixa de 10h a 1Fh. O segundo byte é sempre um caractere imprimível na faixa de 20h a 7Fh. Qualquer par de código de controle que não tiver

[5] vide tabelas em anexo.

função assinalada é ignorado. Se o caractere não imprimível no par estiver na faixa de 00h a 0Fh, então este caractere sozinho será ignorado e o próximo será tratado normalmente

Se o segundo byte de controle não tiver a paridade ímpar, então o par é ignorado. A transmissão redundante do par terá a instrução que o decodificador deverá seguir.

Se o primeiro byte da primeira transmissão do código de controle falhar no teste de paridade, então o byte será inserido na memória corrente como um caractere 7Fh seguido pelo segundo caractere.

Se a primeira transmissão do controle de código passar no teste de paridade, ele é utilizado para o primeiro quadro. Se o próximo quadro contiver uma cópia idêntica do mesmo par, então o código redundante é ignorado. Se o segundo par, no próximo quadro, contiver código de controle diferente , mas também válido este será considerado como código ativo e o receptor esperará pelo próximo par no próximo quadro. Se o primeiro byte do código de controle redundante esperado falhar no teste de paridade e o segundo byte é idêntico ao segundo byte no par imediatamente anterior, então o código redundante é ignorado. Se existirem caracteres imprimíveis no lugar de códigos redundantes, eles são processados normalmente.

Existe um preparação para decodificar o segundo canal de dados. O segundo canal de dados é decodificado com os mesmos controles de códigos e procedimento já descritos. O primeiro byte de cada par de controle de código indica o canal de dados (C1/C2) em que o comando se aplica. Códigos de controle que não são dos canais selecionados pelo usuário e todos os dados subseqüentes relacionados com aquele código de controle são ignorados pelo receptor.

Rejeição de dados - O receptor deve possuir um procedimento para verificar dados. Um receptor rejeitará dados se o dados for inválido ou se o dado não se referir ao canal ou campo selecionado pelo usuário. Dados inválidos são quaisquer dados que falhem no teste de paridade ímpar, ou que passem no teste de paridade, mas não tenham nenhuma função.

Se um caractere imprimível falhar no teste de paridade, um bloco sólido (7Fh) deve ser usado em seu lugar. Além disso, dados válidos podem estar corrompidos de diversas maneiras e podem não ser adequados para visualização na tela. Por exemplo, campos repetidos ou seqüências alteradas de campos são todos possíveis de acontecer.

Receptor deve ignorar dados rejeitados por serem devidos a canais ou campos não selecionados. Entretanto, isto não deve desabilitar o *display*.

Habilitação/desabilitação automática do *display* - O receptor deve prover uma capacidade de habilitação/desabilitação automática para prevenir a exibição de um dado inválido ou incompleto, quando o usuário seleciona o modo *caption*. O *display* deve tornar-se automaticamente habilitado depois de o receptor verificar o dado com descrito anteriormente. O display deve se desabilitar automaticamente quando existir uma detecção sustentada de dados inválidos. O display será reabilitado quando o processo de verificação de dados estiver satisfeito novamente.

PROGRAMAS

Durante o decorrer do estágios foram desenvolvidos programas na linguagem *Assembler*, voltado para o microcontrolador LC8641XX - *SANYO,* os quais foram testados na placa emuladora do referido microcontrolador, usando o PC.

Relógio

No decorrer da Etapa 3, após o estudo das principais funções do referido microcontrolador, e de suas respectivas instruções, o programa do relógio foi o primeiro contato com a programação em *assembler.*

Esse programa, **rel.asm** (vide anexo), simula um relógio com segundos, minutos e horas, utilizando os recursos de interrupção do *timer/counter*, como contador programável de 16-bits (*Timer 0* Modo 2). Anteriormente ao programa, foi elaborado seu fluxograma.

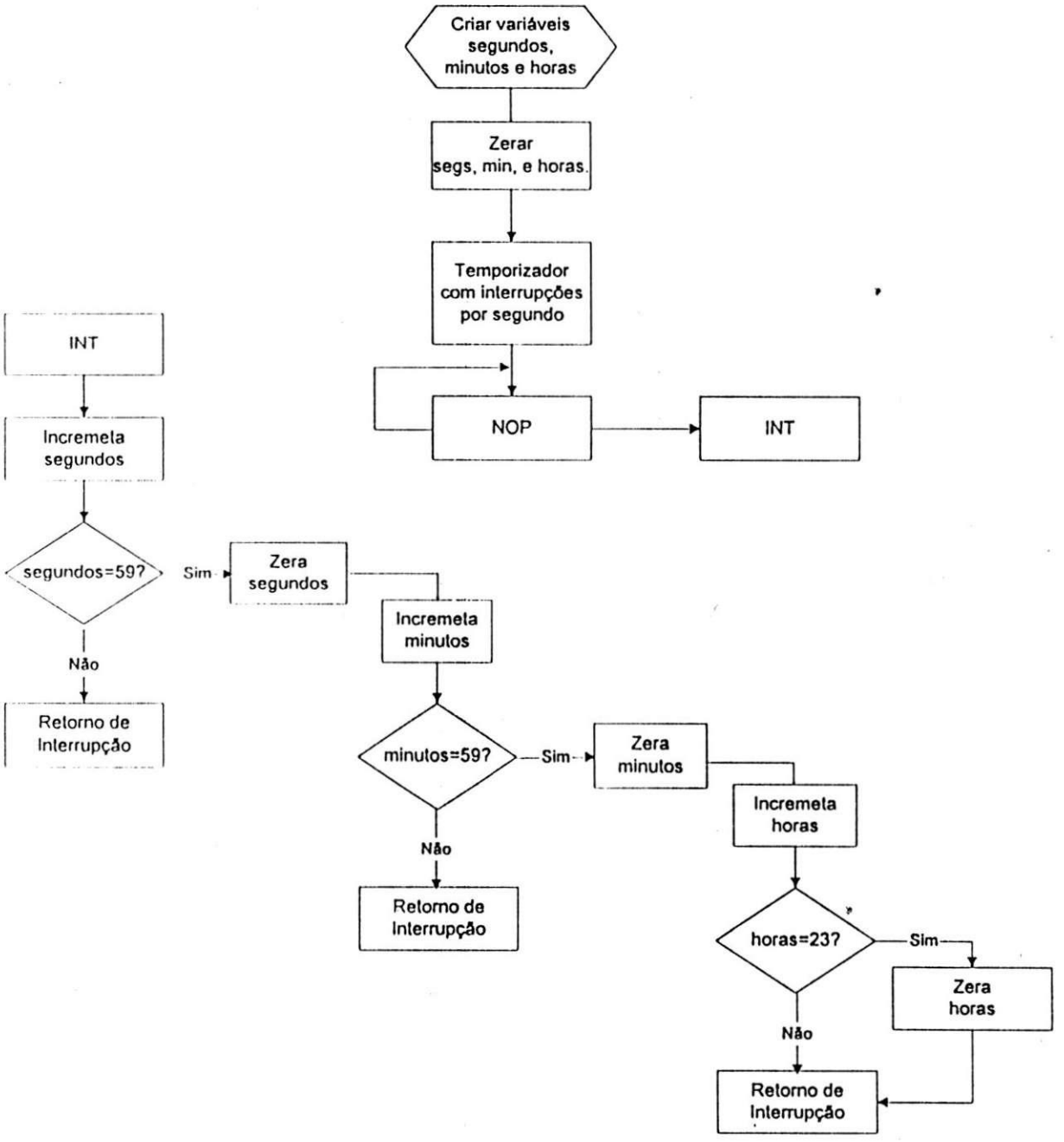


Figura 15: Fluxograma - Relógio

OSD (Escrita)

O programa **tosd.asm** (vide anexo) primeiramente limpa a RAM de vídeo. Em seguida, configura os atributos que determinam a aparência dos caracteres (para uma linha). Foi escolhida um *string* de caracteres a serem impressos na tela do TV e seus atributos foram variados, como cor, tamanho, espaçamento, início e término da linha horizontal, etc.

Protocolo I^2C

Para uma maior familiarização com o protocolo de transmissão serial, foi implementado um *soft* que escreve 4 bytes numa memória E2PROM, **write.asm**, no modo MULTIBYTE. Foi dito anteriormente, utilizamos a memória ST24C01, cujo endereço (físico) foi 001 (b3-b1) do primeiro byte.

Logo após com o programa **read.asm** comprovou-se a correta escrita, como a correta leitura do dado.

Closed Caption (Captura dos Dados)

Como primeiro passo para a implementação do *soft* decodificador de closed *caption*, foi necessário a utilização de um gerador de sinais de TV, com recursos de *closed caption*. Esse sinal deve ser "capturado" e armazenado, para uma posterior análise do código.

O registrador FECR2 (bit0-5) do microcontrolador, ajusta o detector de dados e o detector do sincronismo horizontal, do sinal de vídeo. Inicialmente houve problemas quanto a recepção desse sinal, pois mesmo tendo uma ligação direta (via cabo coaxial) entre o gerador e a placa emuladora, o ajusto é muito sensível a pequenas variações desses nível. Finalmente foi adotado um melhor ajuste para a boa aquisição dos dados (PEDESTAL + 50mv , PEDESTAL - 270mv).

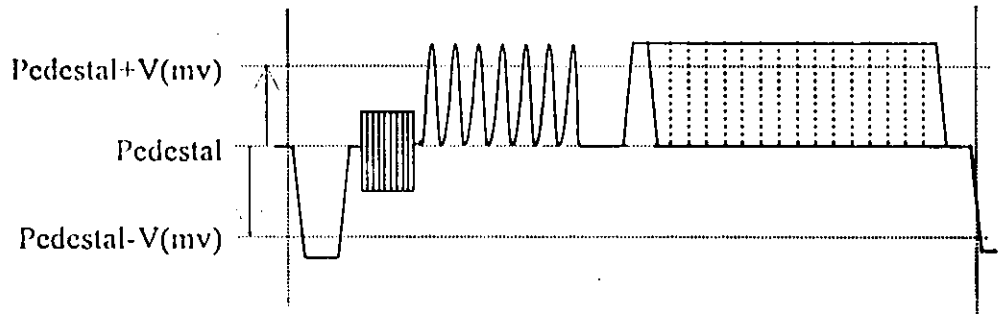


Figura 16: Ajuste do nível do detector de dados e sincronismo horizontal.

Após a detecção da interrupção do *data slicer* e dado um *delay*, dando um tempo suficiente (63 μs) para o reconhecimento do *start bit*, logo em seguida é gravado o conteúdo dos registros CPDH e CPDL diretamente na VRAM. Após o preenchimento total da VRAM a interrupção é desativada. O programa **ccapt.asm** segue em anexo.

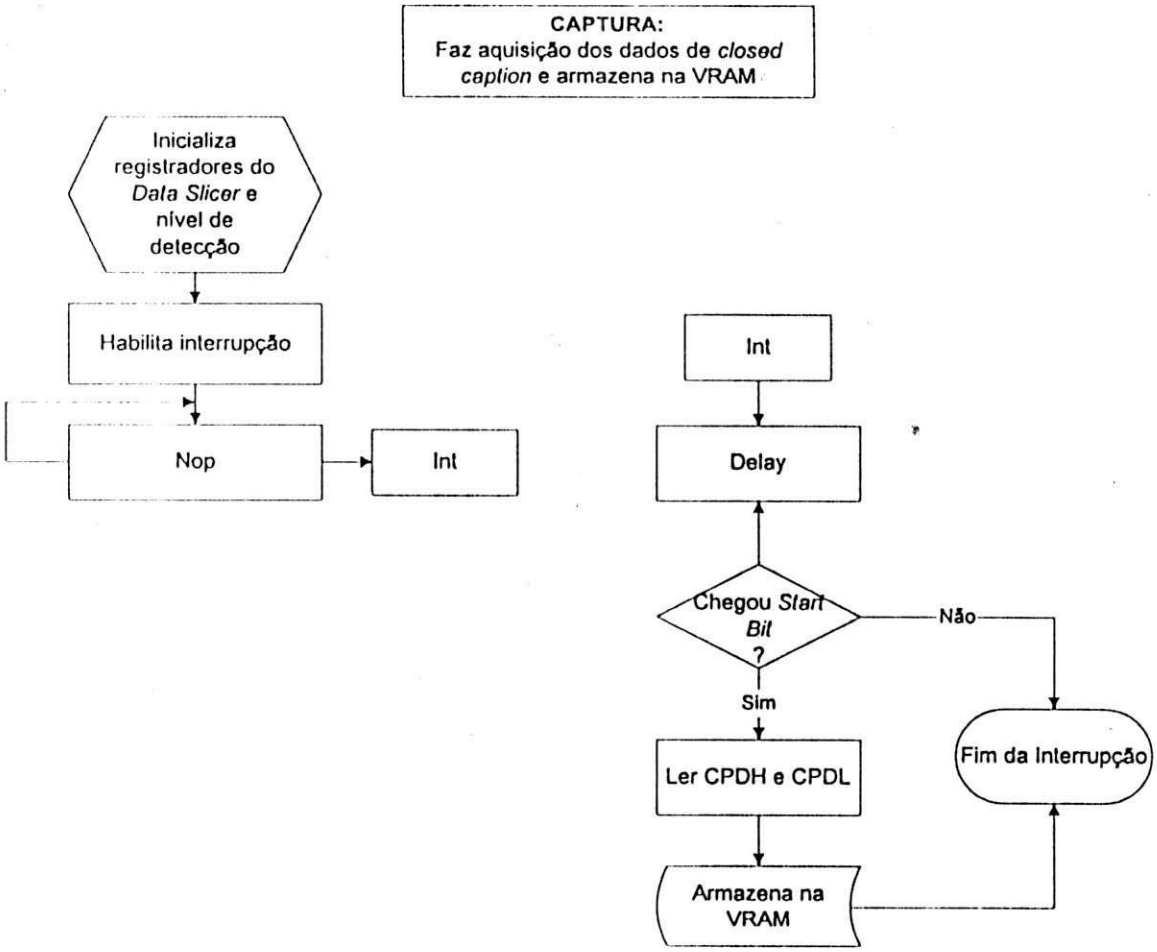


Figura 17: Fluxograma Captura

Estilo Roll-up

Como uma evolução do programa **OSD Escrita,** foi implementado um novo programa (para OSD), a fim de tornar as linhas móveis para cima e escrevendo uma nova *string* na última linha, pretendendo-se assemelhar ao estilo *Roll-up* do Modo *Caption.* Foram utilizados recursos de endereçamento direto e *timer*.

O Fluxograma a seguir se refere ao programa **roll.asm,** que se encontra em anexo.

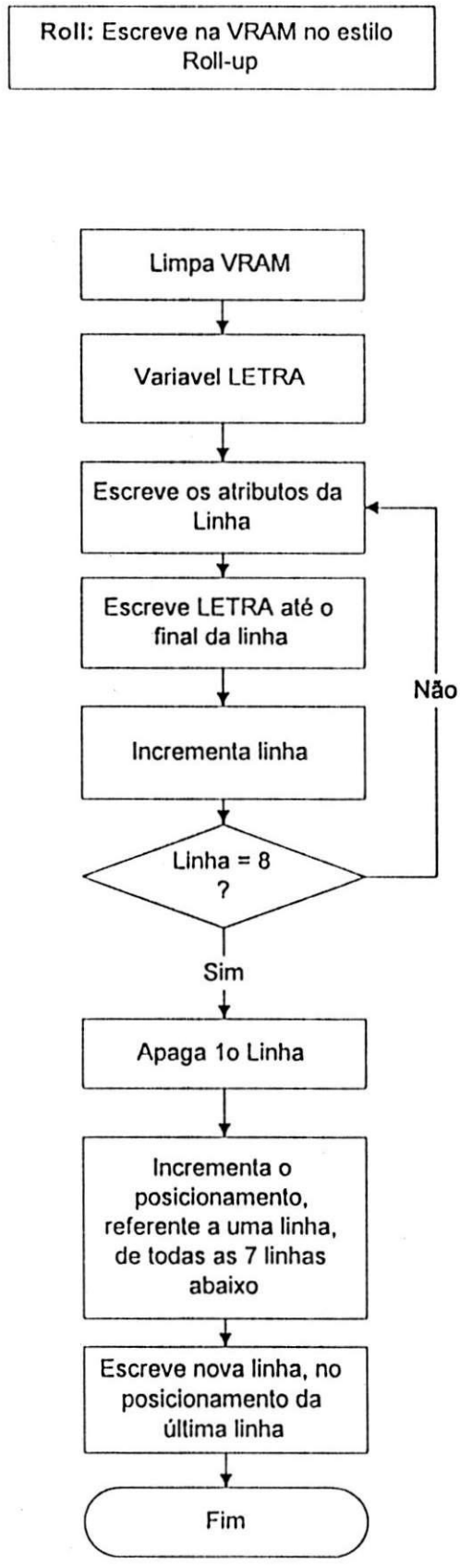


Figura 18: Fluxograma Roll.asm

Decodificador Closed Caption

Como o Brasil segue as tendências mundiais, num futuro próximo, seus aparelhos receptores também deverão estar aptos a decodificar o sinal de *closed caption*. Com esse propósitos o Departamento de Engenharia e Desenvolvimento de Produtos/Imagem desenvolve o *software* a ser utilizados em seus televisores.

Preparação

Por meio de pesquisas feitas na *internet*[6], foi adquirido parte do material bibliográfico, como a norma FCC 47 CFR Part 15, necessário para a realização desse trabalho. Estava também a disposição o manual *For Line 21 Data Service - EIA-608* (já existente nos arquivos da empresa). A partir disso, foi elaborado o fluxograma do programa principal.

Fluxogramas

Apresenta-se aqui uma visão geral do programa, no qual consiste de uma chamada de interrupção, sempre na linha 21 do retraço vertical. O sinal poderá vir no campo 1 ou campo 2, sendo reconhecido via *software*. Toda parte de captura dos dados, análise e execução de ações, está implementado dentro da interrupção. Este tempo de execução total do programa não poderá ser superior a 16,667 ms (1/60 Hz), tempo do retraço vertical, de um campo. Segue em anexo alguns fluxogramas mais detalhados do programa desenvolvido[7].

[6] http://www.fcc.gov

[7] Esses fluxogramas são bastantes próximos do programa, tornando-os bastantes complexos.

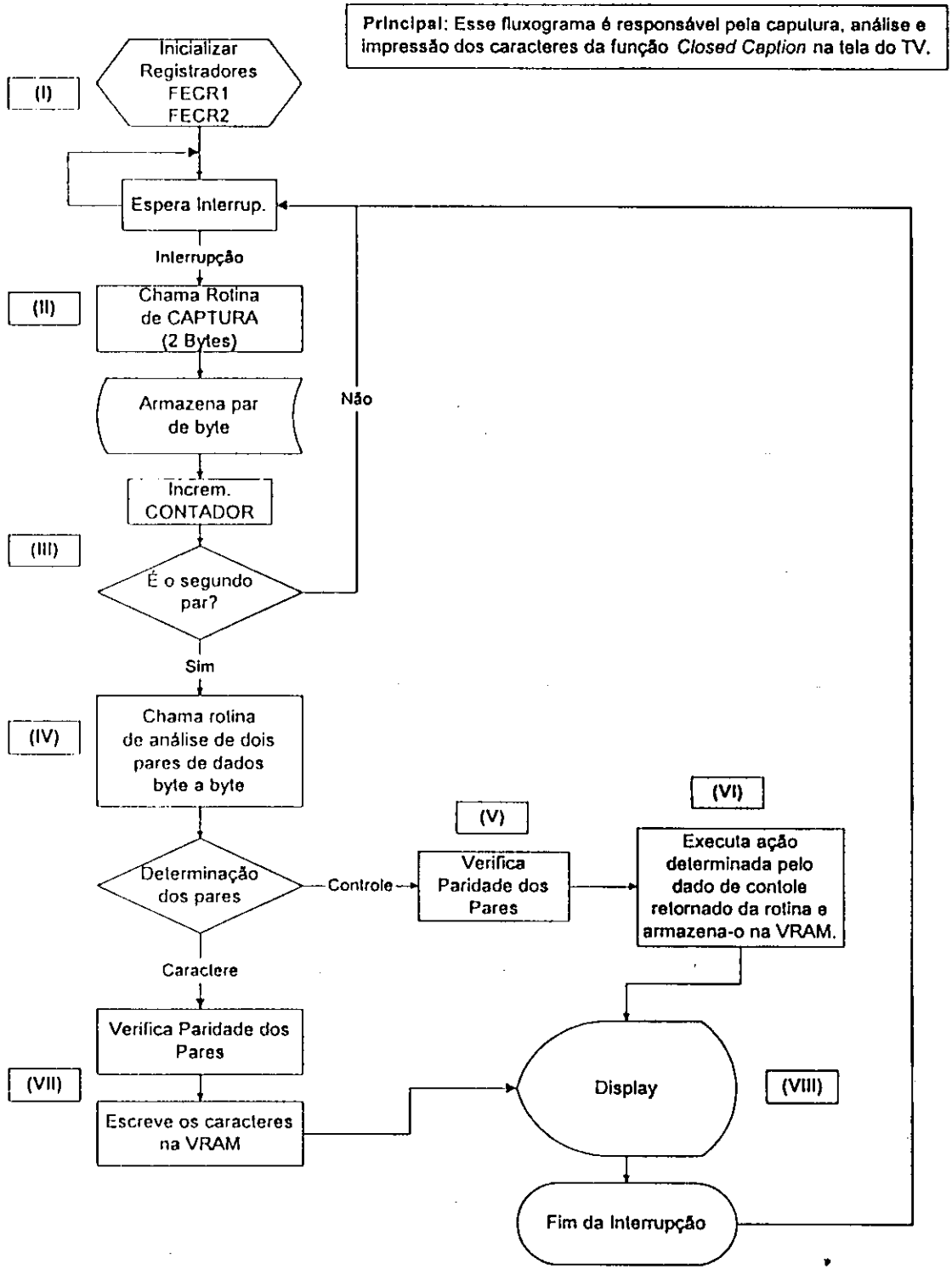


Figura 19: Fluxograma - Decodificador *Closed Caption*

A tabela abaixo, relaciona cada etapa do fluxograma ao referido programa em *assembler*, em anexo.

Nº	PROGRAMA (.asm)
I	teste
II	captura
III	analise
IV	analise
V	controle
VI	exec_con ; miscella ; cont_car
VII	caracter
VIII	esc_vram

Implementação do Soft

Numa primeira tentativa, após a captura de um par de byte, esses seriam analisados e armazenados numa variável temporária. Logo após se executaria a ação determinada ou a impressão do caracter respectivo. Esse idéia foi abandonada, pois observou-se que os pares de controle são enviados duas vezes, para maior segurança da informação, e se perderia tempo na execução do programa. Logo, numa tentativa de melhora o processo, reduzindo o tempo de execução do programa e otimizar linhas de códigos, foi adotado um novo procedimento, o qual se tornou mais viável.

Este consiste na aquisição sempre de dois em dois pares byte (4 bytes), já que os **bytes de controle** são codificados em pares de bytes (ex: 14 29_H). Esta aquisição é feita através da chamada da interrupção *Data Slicer*. Logo em seguida esses pares são analisados e verificados (pela faixa de valores) se são pares de controle ou caracteres.

A partir disto o programa se divide, pois o tratamento para essas duas possibilidades são diferentes. Para o caso dos pares serem **bytes de controle**, esses retornam o resultado de uma ação que estão armazenados numa tabela, podendo ser deste a escrita de um atributo na VRAM (cód. PAC ou MRC), até a preparação de VRAM para um determinado estilo (cód. MISCELLANEOUS). Outro caso, se os pares serem **bytes de caracteres**, estes são escritos diretamente na posição corrente da VRAM. Após a execução desses pares de bytes, o programa sai da interrupção e retorna a execução do programa principal.

Testes

Como primeiro passo, os programas foram testados isoladamente, assumindo valores em suas variáveis de entrada e verificando os resultados nas varáveis de saídas (verificar tabela de variáveis em anexo). Após a retiradas dos erros de compilação e de estrutura do programa, partiu-se para a etapa de junção dos programas e simulação completa da decodificação do sinal. Para realização dos testes dinâmicos, foi necessário a utilização do gerador de sinal *closed caption* no Modo Texto, bem como uma fita de vídeo padrão, gravada com o sinal codificado *closed caption*.

Problemas

Alguns problemas foram observados no decorrer do projeto:

- O não esclarecimento de procedimentos em algumas situações de erros na decodificação, por parte da norma da FCC;
- erros na recepção
- presença de ruído, quando o sinal era gerado pelo vídeo cassete;

Resultados

Superada a etapa dos problemas, verificamos um perfeito sincronismo na decodificação do sinal, como mostra a Figura 20. Foi implementado o programa básico para decodificação do sinal de *closed caption*, para interpretar os dados para o Modo Texto, no canal dados 1.

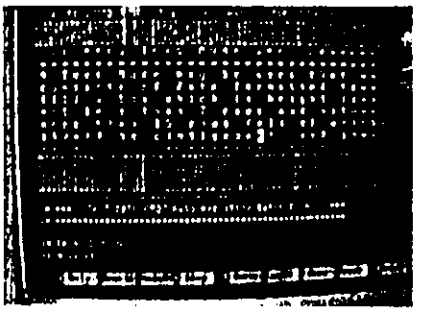


Figura 20: Resultado da simulação do Modo Texto

Num trabalho posterior, será implementado a decodificação para os três estilos do Modo *Caption*, sendo apenas acrescentado as ações de rolagem dos caracteres na tela, para os respectivos estilos.

CONCLUSÃO

De acordo com o cronograma proposto, podemos observar um antecipação dos resultados. Essa fato se deve a extrema dedicação as tarefas, e a um bom plano de trabalho organizado pelo orientador.

No início muitos conceitos sobre TV foram revisados, corrigidos e atualizados com a comprovação na prática. E após a familiarização com o microcontrolador LC8641XX, várias estruturas de programação foram implementadas. Dentro dos programas foram implementadas várias técnicas de endereçamento, armazenamento de dados (como tabelas de dados), área de *buffer*, chamadas de subrotinas usando PUSH e POP, e CALL, interrupções, visto que podemos aplicar a teoria da sala de aulas diretamente num projeto de um *software*, para o mercado.

Outras atividades como participações em reuniões, discussões, processo de implantações da ISO 9001, montagem de protótipos, acompanhamento do processo de desenvolvimento TV's (desde o projeto até sua fabricação), e relacionamento pessoal no ambiente de trabalho, contribuíram para o enriquecimento do aprendizado.

Além do contato direto com vários engenheiros e técnicos experientes na área, tive a oportunidade de manusear vários equipamentos modernos como osciloscópio para TV, gerador de sinais de TV, analisador de espectro, como podemos observar os equipamentos da bancada, na Figura 21.

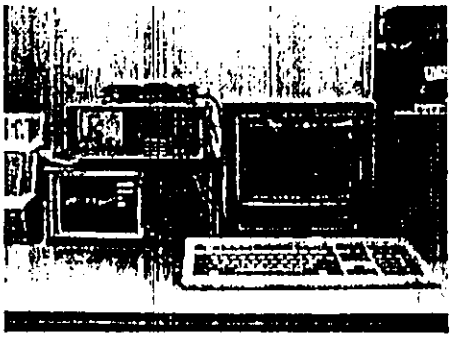


Figura 21: Bancada de trabalho

O trabalho desenvolvido nesse estágio foi de grande contribuição para a empresa, pois em breve os decodificadres *closed caption* serão uma realidade nos televisores SHARP. A realização do estágio nesta respeitável empresa, foi uma grande experiência profissional como complementação da formação acadêmica, podendo assim presenciar todo um processo industrial de grande porte, desde a especificação de produtos até o início da produção.

Figura 22: Estagiário da EDP/I

BIBLIOGRAFIA

- Televisão e Sistemas de Vídeo, Bernard Grob. Ed Guanabara, 1989;
- *CMOS 8-Bit MICROCONTROLLER for Caption-OSD TV set - LC864100 Serie; User's Manual - SANYO Eletric Co., Ltd* - setembro, 1995;
- *EVA86000 Development Tool Manual - SANYO Eletric Co*, Vol I, II, e III;
- *Federal Communications Commission* 47 CFR Part 15.119, outubro 96;
- *Line 21 Data Servide - EIA-608*;
- *Application Notes and Development Tools for 80C51 Microcontroler - Data Handbook - PHILIPS*, 1997;

ANEXOS

REL.ASM

```
    CHIP   LC864132
    JMP    INICIO

    ORG    0060H
INICIO:
    MOV    #12H,OCR      ;SET CLOCK MAIN
    MOV    #00H,20H      ;ENDERECO DOS SEGUNDOS
    MOV    #00H,21H      ;ENDERECO DOS MINUTOS
    MOV    #00H,22H      ;ENDERECO DAS HORAS
    MOV    #06H,T0PRR
    MOV    #0FCH,T0HR
    MOV    #18H,T0LR
    MOV    #0E4H,T0CNT   ;TIMER 0, MODO 2
    MOV    #81H,IE
LOOP:
    NOP
    JMP    LOOP

;*****************************
;INTERRUPCAO
;*****************************

    ORG    0023H
    JMP    RELOGIO

    ORG    0080H
RELOGIO:
    CLR1   T0CNT,3       ;RESET FLAG DA INTERRUP‡ÆO T0HR
    INC    20H
    LD     20H
    BE     #03CH,ZERA_S   ;COMPARA O VALOR DOS SEGUNDOS, SE IGUAL A 59D
    RETI
ZERA_S: MOV    #00H,20H
    INC    21H
    LD     21H
    BE     #03CH,ZERA_M   ;COMPARA O VALOR DOS MINUTOS, SE IGUAL A 59D
    RETI
ZERA_M: MOV    #00H,21H
    INC    22H
    LD     22H
    BE     #018H,ZERA_H   ;COMPARA O VALOR DAS HORAS, SE IGUAL A 23D
    RETI
ZERA_H: MOV    #00H,22H
    RETI
```

TOSD.ASM

```
    CHIP   LC864132
    JMP    START
    ORG    0050H
START:
    MOV    #00000000B,OSDCR2
    SET1   FECR1,7
    MOV    #000H,OSDCR1
    MOV    #10000000B,OSDCR2
;*************************************
;Limpa a VRAM
;*************************************

    MOV    #027H,CLMR
    MOV    #00FH,ACC
L1:   ST     ROWR
L0:   MOV    #000H,VRAM
    DBNZ   CLMR,L0
    MOV    #000H,VRAM
    MOV    #027H,CLMR
    DBNZ   ACC,L1
    MOV    #027H,CLMR
    MOV    #000H,ROWR
L2:   MOV    #000H,VRAM
    DBNZ   CLMR,L2
    MOV    #000H,CLMR
    MOV    #000H,VRAM

;*************************************
;Inicio da Escrita dos Atributos
;*************************************
    MOV    #01000000B,OSDCR1
    MOV    #00000000B,OSDCR2
    MOV    #007H,ROWR
    MOV    #068H,VRAM          ; COL 0
    INC    CLMR                ; COL 1
    MOV    #00000000B,VRAM
    INC    CLMR                ; COL 2
    SET1   OSDCR2,7
    MOV    #00001110B,VRAM     ;MODO osd1,3X4,BACK yellow
    INC    CLMR
    CLR1   OSDCR2,7            ; COL 3
    MOV    #10011100B,VRAM
    INC    CLMR
    SET1   OSDCR2,7            ; COL 4
    MOV    #01110101B,VRAM
    INC    CLMR
    MOV    #11101111B,VRAM     ; COL 5
    MOV    #006H,CLMR
    CLR1   OSDCR2,7
    MOV    #11000111B,OSDCR1

;*********************************
;Escrita do caracter 'a'
;*********************************

    MOV    #041H,VRAM
```

```
        INC   CLMR
        MOV   #0411H,VRAM
        INC   CLMR
        MOV   #0411H,VRAM
        INC   CLMR
        MOV   #0411H,VRAM
        INC   CLMR
        MOV   #0411H,VRAM
        INC   CLMR
        MOV   #0411H,VRAM
        INC   CLMR

P:      JMP   P
        END


        CHIP  LC864132
```

WRITE.ASM

```
;Este programa implementa o protocolo I2c para a escrita de 4 BYTES na
;memória E2PROM, modo MULTIBYTE, necessário colocar o pino MODE/WR em Vih.
;O endereço de acesso a memória é 001 (b3-b1) do primeiro byte.
;Freqüência de transmissao:8,95khz (SCL) - 0,111ms.
;*******************************************************************************


    ;P1,0  -    SCL  BUS
    ;P1,1  -    SDA  BUS

    R1    EQU   0001H
    DVS   EQU   0015H
    ADD1  EQU   0016H
    DT1   EQU   0017H
    DT2   EQU   0018H
    DT3   EQU   0019H
    DT4   EQU   001AH
    CONT  EQU   0010H

    ORG   0000H
    JMP   INICIO

    ORG   0050H
INICIO:
    MOV   #0001H,PSW
    MOV   #10100010B,DVS          ;END. DO DISPOSITIVO
    MOV   #00000101B,ADD1         ;ENDERECO INICIAL DA ESCRITA
    MOV   #0048H,DT1              ;VALOR DO DADO 1
    MOV   #0041H,DT2              ;VALOR DO DADO 2
    MOV   #0052H,DT3              ;VALOR DO DADO 3
    MOV   #0050H,DT4              ;VALOR DO DADO 4
    MOV   #0012H,OCR
    SET1  P1,0            ;SCL=HIGH
    SET1  P1,1            ;SDA=HIGH
    MOV   #00000011B,P1DDR
    MOV   #0001H,P1FCR
    MOV   #0008H,CONT
START:
    NOP
    NOP
    NOP
    CLR1  P1,1            ;SDA=HIGH TO LOW
    NOP
    NOP
    NOP
    MOV   #0015H,R1           ;END1=15H

LP1:
    LD    @R1
    ST    B

LOWSCL:
    CALL  DELAY1
    CLR1  P1,0            ;SCL=LOW
    LD    B
    ROLC
```

```
    ST    B
    LD    PSW
    CALL  TRANS              ;INICIO TRANSMISSAO ESCRITA (DEV SEL)
    CALL  DELAY1
    SET1  P1,0               ;SCL=HIGH
    CALL  DELAY2
    DBNZ  CONT,LOWSCL
    MOV   #008H,CONT
    CLR1  P1,0               ;SCL=LOW
    SET1  P1,1
    CALL  DELAY1
    SET1  P1,0               ;SCL=HIGH (GERA 9o PULSO EM SCL)
    CALL  DELAY1
    CALL  DELAY1
ACK:
    LD    P1                 ;LER ACK (RECONHECE O ACK)
    AND   #00000010B
    BNZ   ACK
    CLR1  P1,1               ;SDA=LOW (SEGURA SDA EM LOW)
    CALL  DELAY1
    CLR1  P1,0               ;SCL=LOW  (LEITURA DO BIT ACK)
    CALL  DELAY1
    INC   R1
    LD    R1
    BNE   #01BH,LP1          ;ULTIMO END. DO DADO +1
    SET1  P1,0
    CALL  DELAY1
    SET1  P1,1               ;SDA=HIGH  (FIM DO BIT DE STOP)

STOP:  BR    STOP

TRANS:
    AND   #10000000B
    BNZ   ONE
    BR    ZERO
ONE:
    SET1  P1,1
    BR    ED
ZERO:
    CLR1  P1,1
ED:   RET

DELAY1: MOV   #00AH,ACC
AQUI:  DBNZ   ACC,AQUI
    RET

DELAY2: MOV   #014H,ACC
QUI:   DBNZ   ACC,QUI
    RET

    END


    CHIP   LC864132
```

READ.ASM

```
;Executa a leitura de um byte  na memória E2PROM
;RANDOM ADDRESS READ
;*****************************************************************************


    ;P1,0 -    SCL  BUS
    ;P1,1 -    SDA  BUS

    R1    EQU   0001H
    R0    EQU   0000H
    DVS   EQU   0015H
    ADD1  EQU   0016H
    DT1   EQU   0017H
    DT2   EQU   0018H
    CONT  EQU   0010H
    CONT1 EQU   0011H
    READ  EQU   0040H

    ORG   0000H
    JMP   INICIO

    ORG   0050H
INICIO:
    MOV   #000H,PSW
    MOV   #10100010B,DVS      ;ENDERECO DO DISPOSITIVO (DEV. SELECT1)
    MOV   #00000101B,ADD1     ;ENDERECO DA E2PROM A SER LIDO
    MOV   #10100011B,DT1      ;ENDERECO DO DISPOSITIVO (DEV. SELECT2)
;   MOV   #0FFH,DT2
    MOV   #012H,OCR
    MOV   #040H,R0
START:
    SET1  P1,0            ;SCL=HIGH
    SET1  P1,1            ;SDA=HIGH
    MOV   #00000011B,P1DDR
    MOV   #000H,P1FCR
    MOV   #008H,CONT
    MOV   #004H,CONT1
    NOP
    NOP
    NOP
    CLR1  P1,1            ;SDA=HIGH TO LOW
    NOP
    NOP
    NOP
    MOV   #015H,R1          ;END1=15H


LP1:
    LD    @R1
    ST    B

LOWSCL:
    CALL  DELAY1
    CLR1  P1,0            ;SCL=LOW
    LD    B
    ROLC
```

```
    ST    B
    LD    PSW
    CALL  TRANS              ;INICIO TRANSMISSAO ESCRITA (DEV SEL)
    CALL  DELAY1
    SET1  P1,0               ;SCL=HIGH
    CALL  DELAY2
    DBNZ  CONT,LOWSCL
    MOV   #008H,CONT
    CLR1  P1,0               ;SCL=LOW
    SET1  P1,1
    CALL  DELAY1
    SET1  P1,0               ;SCL=HIGH (GERA 9o PULSO EM SCL)
    CALL  DELAY1
    CALL  DELAY1
;ACK:
;   LD    P1                 ;LER ACK (RECONHECE O ACK)
;   AND   #00000010B
;   BNZ   ACK
    CLR1  P1,1               ;SDA=LOW (SEGURA SDA EM LOW)
    CALL  DELAY1
    CLR1  P1,0               ;SCL=LOW  (LEITURA DO BIT ACK)
    CALL  DELAY1
    INC   R1
    LD    R1
    BNE   #017H,LP1
    SET1  P1,0               ;SCL=HIGH
    SET1  P1,1               ;SDA=HIGH
    CALL  DELAY1
    CLR1  P1,1               ;SDA=HIGH TO LOW
    CALL  DELAY1
    MOV   #017H,R1           ;END1=15H

LP2:
    LD    @R1
    ST    B

LOWSCL2:
    CALL  DELAY1
    CLR1  P1,0               ;SCL=LOW
    LD    B
    ROLC
    ST    B
    LD    PSW
    CALL  TRANS              ;INICIO TRANSMISSAO ESCRITA (DEV SEL)
    CALL  DELAY1
    SET1  P1,0               ;SCL=HIGH
    CALL  DELAY2
    DBNZ  CONT,LOWSCL2
    MOV   #008H,CONT
    CLR1  P1,0               ;SCL=LOW
    SET1  P1,1
    CALL  DELAY2
    SET1  P1,0               ;SCL=HIGH (GERA 9o PULSO EM SCL)
    CALL  DELAY2
ACK2:
    LD    P1                 ;LER ACK (RECONHECE O ACK)
    AND   #00000010B
    BNZ   ACK2
    CALL  DELAY2
```

```
    SET1   P1,1             ;SDA=HIGH  (FIM DO BIT DE STOP)

LOWSCL3:                    ;INICIO DA AQUISICAO DOS BYTE

    CALL   DELAY2
    CLR1   P1,0             ;SCL=LOW
    CALL   DELAY1
    SET1   P1,0             ;SCL=HIGH
    CALL   DELAY1


    LD     P1
    AND    #00000010B
    CALL   DECIDE           ;VERIFICA O BIT DE LEITURA
    ROLC
    ST     @R0

    DBNZ   CONT,LOWSCL3
    MOV    #008H,CONT
    RORC
    ST     @R0
    INC    R0
    DBNZ   CONT1,MANDACK
; XXXX MANDA NOT ACK XXXXXX
    CALL   DELAY2
    CLR1   P1,0             ;SCL=LOW
    CALL   DELAY2
    SET1   P1,1             ;SDA=HIGH (NOT ACK)
    CALL   DELAY2
    SET1   P1,0             ;SCL=HIGH
    CALL   DELAY2

; XXXX BIT DE STOP XXXXXX
    CALL   DELAY2
    CLR1   P1,0             ;SCL=LOW (INICIO DO BIT DE STOP)
    CALL   DELAY2
    CLR1   P1,1             ;SDA=LOW
    CALL   DELAY2
    SET1   P1,0             ;SCL=HIGH
    CALL   DELAY2
    SET1   P1,1             ;SDA=HIGH


STOP:  BR     STOP

TRANS:
    AND    #10000000B
    BNZ    ONE
    BR     ZERO
ONE:
    SET1   P1,1
    BR     ED
ZERO:
    CLR1   P1,1
ED:    RET

DELAY1: MOV    #00AH,ACC
AQUI:  DBNZ    ACC,AQUI
    RET
```

```
DELAY2: MOV    #014H,ACC
QUI:   DBNZ   ACC,QUI
       RET


DECIDE: BN     ACC,1,CLEAR
       XCH    @R0
       SET1   ACC,0
       BR     FIM
CLEAR: XCH    @R0
       CLR1   ACC,0
FIM:   RET

MANDACK:CLR1   P1,0              ;SCL=LOW (MANDA O ACK)
       CALL   DELAY2
       CLR1   P1,1              ;SDA=LOW
       CALL   DELAY2
       SET1   P1,0              ;SCL=HIGH
       CALL   DELAY2
       CLR1   P1,0              ;SCL=LOW
       CALL   DELAY2
       SET1   P1,1
       JMP    LOWSCL3

       END
```

CCAPT.ASM

```
    CHIP   LC864132
    JMP    START

DECREMEN     EQU    045H

    ORG    0050H
START:

;*******************
;LIMPA LINHA DA VRAM
;*******************

    MOV    #012H,OCR
    MOV    #0FFH,T0PRR
    MOV    #00FH,T0LR
    MOV    #00000100B,PPCR
    MOV    #00000000B,OSDCR2
    MOV    #000H,OSDCR1
    MOV    #10000000B,OSDCR2
    MOV    #027H,CLMR
    MOV    #00FH,ACC
L1:   ST     ROWR
L0:   MOV    #000H,VRAM
    DBNZ   CLMR,L0
    MOV    #000H,VRAM
    MOV    #027H,CLMR
    DBNZ   ACC,L1
    MOV    #027H,CLMR
    MOV    #000H,ROWR
L2:   MOV    #000H,VRAM
    DBNZ   CLMR,L2
    MOV    #000H,CLMR
    MOV    #000H,ROWR
    MOV    #000H,VRAM

;*****************************************
;INICIALIZACAO DE PARAMETROS DO DATA SLICE
;*****************************************

    MOV    #10110000B,FECR1
    MOV    #00000001B,FECR2      ;Nível do Pedestal
    MOV    #00001000B,VRCR
    MOV    #00000000B,CPDL
    MOV    #00000000B,CPDH
    MOV    #00100000B,IP
    SET1   IE,7                  ;Habilita interrupções

P:    BR     P

;*********************************
;Interrupção
;*********************************

    ORG    03BH
```

```
    JMP   0FAH

    ORG   0FAH


VOLTA: LD    VRCR
    AND   #01110000B
    BZ    VOLTA
VOLTA1: LD    VRCR
    AND   #00000001B
    BZ    VOLTA1
    MOV   #01000000B,T0CNT
FICA:  BN    T0CNT,1,FICA
    CLR1  T0CNT,6
    CLR1  CPDH,7
    CLR1  CPDL,7
    LD    CPDH
    ST    VRAM
    LD    CPDL
    INC   CLMR
    ST    VRAM
    INC   CLMR
    LD    CLMR
    MOV   #000H,VRCR
    CLR1  FECR1,6
    BE    #028H,XXXX
    BR    WAIT
FIM:   CLR1  FECR1,5
    BR    WAIT
XXXX:  MOV   #000H,CLMR
    INC   ROWR
    LD    ROWR
    AND   #00010000B
    BNZ   FIM
WAIT:  RETI
```

ROLL.ASM

```
    CHIP   LC864132

DECREMEN      EQU    004FH
POSICAO       EQU    004EH
LETRA         EQU    004DH
COLUNA        EQU    004CH

    JMP    START
    ORG    0050H
START:

    MOV    #00000000B,OSDCR2
    SET1   FECR1,7
    MOV    #000H,OSDCR1
    MOV    #10000000B,OSDCR2
    MOV    #027H,CLMR
    MOV    #00FH,ACC
L1:   ST     ROWR

L0:   MOV    #000H,VRAM
    DBNZ   CLMR,L0
    MOV    #000H,VRAM
    MOV    #027H,CLMR
    DBNZ   ACC,L1
    MOV    #027H,CLMR
    MOV    #000H,ROWR

L2:   MOV    #000H,VRAM
    DBNZ   CLMR,L2
    MOV    #000H,CLMR
    MOV    #000H,VRAM
    MOV    #11000111B,OSDCR1

    MOV    #00000000B,OSDCR2

    MOV    #007H,ROWR
    MOV    #000H,HDSPR
    MOV    #008H,DECREMEN
    MOV    #068H,POSICAO
    MOV    #041H,LETRA                ;CARACTER 'A'

NOVA:  LD     POSICAO
    ST     VRAM              ; COL 0
    INC    CLMR              ; COL 1
    MOV    #00000000B,VRAM
    INC    CLMR              ; COL 2
    SET1   OSDCR2,7
    MOV    #00110100B,VRAM       ;MODO TEXTO,1X2,BACK
    INC    CLMR
    CLR1   OSDCR2,7          ; COL 3
    MOV    #10010000B,VRAM
    INC    CLMR
    SET1   OSDCR2,7          ; COL 4
    MOV    #01111111B,VRAM
    INC    CLMR
    MOV    #00111010B,VRAM       ; COL 5
```

```
    INC   CLMR
    MOV   #11011000B,VRAM      ; COL 6
    MOV   #007H,CLMR
    CLR1  OSDCR2,7
    LD    CLMR
    ST    COLUNA

L:  LD    LETRA                ;Escreve 'a' ate o final da linha
    ST    VRAM
    LD    COLUNA
    ST    CLMR
    INC   COLUNA
    LD    COLUNA
    BNE   #28H,L

    INC   LETRA                ;Escreve 'b' na próxima linha
    INC   ROWR
    LD    POSICAO
    ADD   #00DH
    ST    POSICAO
    MOV   #000H,CLMR
    DBNZ  DECREMEN,NOVA        ;escreve ate 8 linhas

    MOV   #007H,ROWR
    MOV   #007H,CLMR

LIMPA: MOV  #000H,VRAM         ;APAGA primeira linha
    INC   CLMR
    LD    CLMR
    BNE   #028H,LIMPA

;*****************************
    MOV   #012H,OCR


    MOV   #00100000B,T0CNT

    MOV   #038H,T0PRR          ;TEMPO DE 1s (250 dec)
    MOV   #0FEH,T0HR
    MOV   #0FFH,T0LR
    MOV   #11100000B,T0CNT

VOLTA: BN   T0CNT,3,VOLTA
;*****************************



    INC   ROWR                 ;INCREMENTA LINHAS TV (COLUNA 0)
    MOV   #000H,CLMR
RETURN: LD  VRAM
    SUB   #00DH
    ST    VRAM
    INC   ROWR
    LD    ROWR
    AND   #00FH
    BNE   #00FH,RETURN


    MOV   #00EH,ROWR
```

```
    MOV   #0001H,CLMR
    LD    VRAM              ;ULTIMO VALOR DE POSICAO
    ADD   #00D1H
    MOV   #0071H,ROWR
    ST    VRAM
    MOV   #007FH,CLMR
XXX: MOV  #058H,VRAM        ;ESCREVE 'X' NA NOVA POSICAO (ULTIMA LINHA)
    INC   CLMR
    LD    CLMR
    BNE   #028H,XXX


;**********************************
;     BR    BACK

P:  JMP   P
    END
```

TESTE.ASM

```
;Programa de decodificação do sinal de closed caption
;*****************************************************************************
        CHIP   LC864132


        DSEG


R0:         DS    1
R1:         DS    1
R2:         DS    1
R3:         DS    1
AREA:       DS    4     ;VARIAVEL DOS PARES
VILMA0:     DS    1     ;VARIAVEL DE CONTROLE
VILMA1:     DS    1     ;VARIAVEL DA PARIDADE
DEFINITIVA: DS    4     ;INFORM. DO PAR ESCOLHIDO OU DOS CARACTERES IMPRIMIVEIS
LINHA:      DS    1     ;INFORMACAO DA LINHA
COLUNA:     DS    1     ;INFORMACAO DA COLUNA
BOTTOM_LINE: DS   1
TAB1:       DS    9     ;TABELA DA PARRIDADE
TAB_LINHA:  DS    8     ;TABELA INDICADORA DA LINHA
TAB_MRC:    DS    7
DECREMEN:   DS    1
INCREMEN:   DS    1
MIS:        DS    1
STACK:      DS    16

        CSEG

        ORG   000H
        JMP   START

        ORG   03BH
        JMP   INT_SLICE

        ORG   50H

START:
        MOV   #STACK,SP

;TABELA DE PARIDADE

        MOV   #00H,TAB1
        MOV   #08H,TAB1+1
        MOV   #0CH,TAB1+2
        MOV   #02H,TAB1+3
        MOV   #0AH,TAB1+4
        MOV   #0EH,TAB1+5
        MOV   #03H,TAB1+6
        MOV   #0BH,TAB1+7
        MOV   #0FH,TAB1+8

;TABELA INDICADORA DA LINHA

        MOV   #00BH,TAB_LINHA
        MOV   #001H,TAB_LINHA+1
```

```
        MOV   #003H,TAB_LINHA+2
        MOV   #00CH,TAB_LINHA+3
        MOV   #00EH,TAB_LINHA+4
        MOV   #005H,TAB_LINHA+5
        MOV   #007H,TAB_LINHA+6
        MOV   #009H,TAB_LINHA+7

;TAB_MRC

        MOV   #00000111B,TAB_MRC
        MOV   #00000010B,TAB_MRC+1
        MOV   #00000001B,TAB_MRC+2
        MOV   #00000011B,TAB_MRC+3
        MOV   #00000100B,TAB_MRC+4
        MOV   #00000110B,TAB_MRC+5
        MOV   #00000101B,TAB_MRC+6


    ;*****************************************
    ;INICIALIZACAO DE PARAMETROS DO DATA SLICE
    ;*****************************************

        INCLUDE LIM_VRAM.ASM

        MOV   #000H,COLUNA
        MOV   #000H,LINHA
        MOV   #000H,MIS          ;NAO RECEBEU UM MISCELLANEOUS
        MOV   #012H,OCR
        MOV   #10110000B,FECR1
        MOV   #00000001B,FECR2     ;NÍVEL DE DETEÇÃO
        MOV   #000H,PSW          ;CONFIGURA BANK0
        CLR1  VRCR,0
        MOV   #080H,OSDCR1
        MOV   #000H,OSDCR2
        MOV   #020H,HDSPR
        MOV   #00000100B,PPCR
        MOV   #00100000B,IP      ;PRIORIDADE PARA INT SLICE
        MOV   #0FFH,T0PRR        ;CONFIG CONT DE TEMPO APOS START BIT
        MOV   #00FH,T0LR
        MOV   #AREA,R0
        MOV   #002H,DECREMEN         ;CONTADOR PRINCIPAL DE 2 PARES
        MOV   #000H,INCREMEN

        SET1  IE,7

WAIT_INT:   NOP
        BR    WAIT_INT


        INCLUDE CAPTURA.ASM

        INCLUDE ANALISE.ASM

        INCLUDE CONT_CAR.ASM

        INCLUDE CONTROLE.ASM

        INCLUDE EXEC_CON.ASM
```

```
        INCLUDE MISCELLA.ASM

        INCLUDE CARACTER.ASM

        INCLUDE ESC_VRAM.ASM

XXX:
;PP:      BR    PP
        SET1  OSDCR1,7
        MOV   #AREA,R0
        RETI

        END
```

CAPTURA.ASM

```
INT_SLICE:    CLR1   MIS,1
              MOV    #011H,ACC

ESPERE:       DBNZ   ACC,ESPERE
              BN     VRCR,0,OUT_INT
              MOV    #012H,DECREMEN

FICA:         DBNZ   DECREMEN,FICA
              LD     CPDH
              ST     @R0              ;R0=AREA
              INC    R0
              LD     CPDL
              ST     @R0
              INC    R0
              INC    INCREMEN
              LD     INCREMEN
              BE     #002H,PULE_JUMP
              MOV    #AREA+2,R0

OUT_INT:      MOV    #000H,VRCR       ;LIMPA BIT DE START
              CLR1   FECR1,6
              RETI

PULE_JUMP:    MOV    #000H,VRCR
              CLR1   FECR1,6
              MOV    #000,INCREMEN
              MOV    #AREA,R0
              MOV    #002H,DECREMEN

;             bro    XXX              ;TESTE
;****DAQUI PARA BAIXO E' ANALISE.ASM
```

ANALISE.ASM

```
;Subrotina de decisão se o par de byte é de controle (valido ou não) ou
;caractere.

;******TESTE************************

;INT_SLICE:
;       MOV   #094H,AREA
;       MOV   #0ABH,AREA+1
;       MOV   #094H,AREA+2
;       MOV   #0ABH,AREA+3
;       MOV   #000H,LINHA
;       MOV   #000H,COLUNA
;       MOV   #000,INCREMEN
;       MOV   #AREA,R0
;       MOV   #002H,DECREMEN

;*******************



RICK:     LD    @R0             ;AREA+1=CPDH(PRIM. PAR)
          AND   #070H
          SUB   #010H           ;VERIF. SE CPDH ESTA NA FAIXA 10H-1FH
          BNZ   FORA_FAIXA
          SET1  VILMA0,7        ;PRIM. BYTE VALIDO
          SET1  VILMA0,6
          BR    PROX

FORA_FAIXA:  CLR1  VILMA0,7     ;PRIM. BYTE INVALIDO
          LD    @R0
          AND   #070H
          SUB   #020H
          BP    PSW,7,WWWW      ;VERIFICA SE CPDH <20H
          SET1  VILMA0,6
          BR    PROX

WWWW:     CLR1  VILMA0,6

PROX:     INC   R0              ;CPDL
          LD    @R0
          SUB   #020H
          BP    PSW,7,BYTE2_INVAL   ;VERIFICA SE CDPL <20H
          SET1  VILMA0,5        ;SEG. BYTE INVALIDO
          BR    BY_PASS

BYTE2_INVAL:  CLR1  VILMA0,5
BY_PASS:  INC   R0
          LD    DECREMEN
          BE    #001H,ABANDONA

          LD    VILMA0          ;ROTACIONA 3X PARA DIREITA
          ROR
          ROR
```

```
        ROR
        ST    VILMA0

ABANDONA:    DBNZ   DECREMEN,RICK        ;DECREMENTA SE NAO ZERO
        LD    VILMA0


;************************
                        ;VERIFICA SE OS DOIS PARES
                        ;SAO DE CONTROLE OU CARACTERE
                        ;OU CONTROLE+CARACTERE

        BN    VILMA0,4,J11
PAR_CONT:    SET1   VILMA0,1
        BR    SEG

J11:      BN    VILMA0,3,PAR_CONT
        CLR1   VILMA0,1

SEG:
        BN    VILMA0,7,J22
CON:      SET1   VILMA0,0
        BR    DECIDE

J22:      BN    VILMA0,6,CON
        CLR1   VILMA0,0

DECIDE:     LD    VILMA0          ;LE RESULTADO EM VILMA0,1-0
        AND   #003H
        bzo   CARACTERES        ;OS PARES SAO DE CARACTERES
        beo   #003H,CONTROLE      ;OS PARES SAO DE CONTROLE
        beo   #001H,CARAC_CONT      ;OS PARES SAO CARACT_CONT
        bro   CONT_CARAC        ; OU VICE-VERSA


;**********SOMENTE PARA TESTES
;CARACTERES:   BR    CARACTERES
;CARAC_CONT:   BR    CARAC_CONT
;CONT_CARAC:   BR    CONT_CARAC
;CONTROLE:     BR    CONTROLE
;************************
```

CONT_CAR.ASM

```
CONT_CARAC:   bno   MIS,0,FORA
        SET1  MIS,1             ;INDICA P/ MISCELLA
        LD    AREA              ;CARREGA CONTROLE(PRIM BYTE)
        BP    PSW,0,N_IGNORA    ;TESTA PARIDADE
        BR    IG_2
N_IGNORA:   LD    AREA+1            ;CARREGA CONTROLE(SEG BYTE)
        BP    PSW,0,VALIDO      ;TESTA PARIDADE
IG_2:     MOV   #000H,DEFINITIVA     ;SE UM OU OUTRO NAO TIVER PAR.
        MOV   #000H,DEFINITIVA+1   ;IGNORA-SE OS DOIS
        BR    AREA_2

VALIDO:     LD    AREA
        ST    DEFINITIVA
        LD    AREA+1
        ST    DEFINITIVA+1

AREA_2:     LD    AREA+2
        BP    PSW,0,N_B_SOL
        MOV   #07FH,DEFINITIVA+2
        BR    AREA_3

N_B_SOL:    CLR1  ACC,7
        ST    DEFINITIVA+2

AREA_3:     LD    AREA+3
        BP    PSW,0,N_B_SOL2
        MOV   #07FH,DEFINITIVA+3
        BR    PULE_DEF

N_B_SOL2:   BP    VILMA0,5,TA_NA_FAIXA
        MOV   #000H,DEFINITIVA+3
        BR    PULE_DEF

TA_NA_FAIXA:  CLR1  ACC,7
        ST    DEFINITIVA+3

PULE_DEF:   bro   EXEC_CON
```

CONTROLE.ASM

```
                        ;INICIO DA VERIFICACAO DA
                        ;PARIDADE DE CADA PAR DE CONTROLE
CONTROLE:

        MOV   #002H,DECREMEN      ;INICIA CONTADOR DE 2 PARES
        MOV   #AREA+1,R0
        MOV   #AREA,R1

PAR_2:    LD    @R0
        BP    PSW,0,PARITY0       ;VERIFICA PARIDADE EM CPDL
        CLR1  VILMA1,5            ;PAR IGNORADO "00" NO FLAG
        CLR1  VILMA1,4            ;RELATIVO A CPDL
        BR    KIKO

PARITY0:    LD    @R1
        BP    PSW,0,PARITY1       ;VERIFICA PARIDADE EM CPDH
        SET1  VILMA1,5            ;PAR BLOCO SOLIDO
        CLR1  VILMA1,4
        BR    KIKO

PARITY1:    SET1  VILMA1,5          ;PAR OK "11"
        SET1  VILMA1,4

KIKO:       INC   R0
        INC   R0              ;INCREMENTA PARA AREA 2 E
        INC   R1              ;AREA 3
        INC   R1
        LD    VILMA1
        ROR
        ROR
        ST    VILMA1          ;ROTACIONA PARA SEG. PAR
        DBNZ  DECREMEN,PAR_2
        LD    AREA            ;VERIF. SE AREA0=AREA2
        BNE   AREA+2,DIFERENTES
        LD    AREA+1          ;VERIFICA SE AREA1=AREA3
        BNE   AREA+3,DIFERENTES
        CLR1  VILMA1,7        ;SAO IGUAIS
        BR    FIM_PARITY

DIFERENTES:   SET1  VILMA1,7          ;SAO DIFERENTES

;****
                        ;COMPARA COM A TABELA
                        ;DE PARIDADE

FIM_PARITY:   LD    VILMA1              ;CONTROLE+CONTROLE
        AND   #00FH
        BE    #000H,BLOCO_SOLIDO
        BE    #00001000B,BLOCO_SOLIDO
        BE    #00001100B,SEG_PAR
        BE    #00000010B,BLOCO_SOLIDO
        BE    #00001010B,BLOCO_SOLIDO
        BE    #00001110B,SEG_PAR      ;EH FODA
        BE    #00000011B,PRIM_PAR
        BE    #00001011B,PRIM_PAR
        BE    #00001111B,DECISAO      ;DECISAO
```

```
BLOCO_SOLIDO:  MOV    #07FH,DEFINITIVA          ;ASSUME BLOCO SOLIDO
        MOV    #000H,DEFINITIVA+1
        MOV    #000H,DEFINITIVA+2          ;IGNORA REGISTRADORES
        MOV    #000H,DEFINITIVA+3          ;CASO CONT_CONT
        bro    ESC_VRAM

PRIM_PAR:      LD     AREA                ;ASSUME PRIM_PAR
        ST     DEFINITIVA
        LD     AREA+1
        ST     DEFINITIVA+1
        MOV    #000H,DEFINITIVA+2          ;IGNORA REGISTRADORES
        MOV    #000H,DEFINITIVA+3          ;CASO CONT_CONT
        BR     EXEC_CON

SEG_PAR:       LD     AREA+2              ;ASSUME SEGUN_PAR
        ST     DEFINITIVA
        LD     AREA+3
        ST     DEFINITIVA+1
        MOV    #000H,DEFINITIVA+2          ;IGNORA REGISTRADORES
        MOV    #000H,DEFINITIVA+3          ;CASO CONT_CONT
        BR     EXEC_CON

DECISAO:       BN     VILMA1,7,PRIM_PAR          ;VERIFICA SE OS PARES
        MOV    #001H,DECREMEN             ;SAO IGUAIS
        BR     SEG_PAR


;**********SOMENTE PARA TESTES
;EXEC_CON:        BR     EXEC_CON
;**********************
```

EXEC_CON.ASM

```
;Esta rotina verifica se o par de controle escolhido é MID ROW CODE,
;MISCELLANEOUS ou PAC
;localiza a posição do CPDL em cada tabela
;executa a ação indicada pelo conteúdo da tabela

EXEC_CON:
        CLR1   DEFINITIVA,7        ;LIMPA BITS DE PARIDADE
        CLR1   DEFINITIVA+1,7
        LD     DEFINITIVA
        BNE    #011H,T_14H         ;TESTA SE CPDH EH 11H
        LD     DEFINITIVA+1
        AND    #0F0H
        beo    #020H,COD_MRC
        BR     COD_PAC

T_14H:      BE     #014H,T_CPDL        ;TESTA SE CPDH EH 14H
        BNE    #017H,COD_PAC       ;TESTA SE CPDH EH 17H

T_CPDL:     LD     DEFINITIVA+1        ;TESTA SE CPDL ESTA 20-2FH
        AND    #0F0H
        beo    #020H,COD_MISC
        BR     COD_PAC

;**************************************
;TRATAMENTO DO CODIGO MID ROW CODES
;**************************************

COD_MRC:    LD     DEFINITIVA+1        ;O COD. EH MID ROW CODES
        SUB    #020H
TAB_COR:    BN     ACC,0,PAR
        SUB    #001H               ;COR COM UNDERLINES
        ROR
        BE     #007H,ITAL_UND      ;VERIFICA SE EH ITALCO COM UNDERLINE
        ADD    #TAB_MRC
        ST     R0
        LD     @R0                 ;LE O CONTEUDO DA TABELA
        SET1   ACC,4               ;HABILITA UNDERLINE
        ST     DEFINITIVA          ;RESULTADO DEFINITVO PARA VRAM
        MOV    #000H,DEFINITIVA+1
        bro    ESC_VRAM
ITAL_UND:   SET1   DEFINITIVA,5        ;COR COM ITALICO UNDERLINE
UND:        SET1   DEFINITIVA,4        ;COR COM UNDERLINE
        MOV    #000H,DEFINITIVA+1
        bro    ESC_VRAM

PAR:        ROR                    ;INICIO DOS NOT UNDERLINES
        BE     #007H,UND
        ADD    #TAB_MRC
        ST     R0
        LD     @R0
        ST     DEFINITIVA          ;ARMAZENA O CONTEUDO DE
        MOV    #000H,DEFINITIVA+1
        bro    ESC_VRAM            ;TAB_MRC + R

;**********************************
```

```
;TRATAMENTO DO CODIGO PAC
;*********************************

COD_PAC:    LD    DEFINITIVA+1
            AND   #0F0H
            BE    #040H,END_4XH
            BE    #060H,END_6XH
            BE    #050H,END_5XH
            BE    #070H,END_7XH
            JMPF  XXX             ;SEM FUNCAO

END_4XH:    LD    DEFINITIVA
            AND   #00FH
            ADD   #TAB_LINHA
            ST    R0
            LD    @R0
            ST    LINHA
            LD    DEFINITIVA+1
APROV_LIN:  SUB   #040H           ;APROVEITA TAB_LINHA
            BE    #00EH,WHITE_ITALIC
            BNE   #00FH,TAB_COR   ;TABELA DE CORES
            MOV   #00110111B,DEFINITIVA  ;WHITE ITALIC UNDERLINE
            MOV   #000H,DEFINITIVA+1
            bro   ESC_VRAM
WHITE_ITALIC: MOV   #00100111B,DEFINITIVA
            MOV   #000H,DEFINITIVA+1
            bro   ESC_VRAM

END_6XH:    LD    DEFINITIVA
            AND   #00FH
            bco   #00AH,XXX       ;SE CPDL=10D, CODIGO INVALIDO
            ADD   #TAB_LINHA
            ST    R0
            LD    @R0
            ADD   #001H
            ST    LINHA
            LD    DEFINITIVA+1
            SUB   #020H
            BR    APROV_LIN

END_5XH:    LD    DEFINITIVA
            AND   #00FH
            ADD   #TAB_LINHA
            ST    R0
            LD    @R0
            ST    LINHA
            LD    DEFINITIVA+1
APROV_COL:  SUB   #050H
            BN    ACC,0,PAR_IND
            SUB   #001H
            ROL
            ST    R0
            MOV   #00010111B,DEFINITIVA   ;ASSUME COR PADRAO COM UNDERLINE
            BR    PULE_PAR_IND
PAR_IND:    ROL
            MOV   #00000111B,DEFINITIVA   ;ASSUME COR PADRAO SEM UNDERLINE
PULE_PAR_IND: BN    MIS,0,NAO_CHEGOU
            LD    COLUNA
```

```
        ST    CLMR
        INC   CLMR
        MOV   #020H,VRAM
        DBNZ  R0,PULE_PAR_IND
        LD    CLMR
        ST    COLUNA
        MOV   #000H,DEFINITIVA+1
        bro   ESC_VRAM
NAO_CHEGOU:   bro   XXX          ;CASO AINDA NAO TENHA MISCELLANEOUS

END_7XH:      LD    DEFINITIVA
        AND   #00FH
        beo   #00AH,XXX
        ADD   #TAB_LINHA
        ST    R0
        LD    @R0
        ADD   #001H
        ST    LINHA
        LD    DEFINITIVA+1
        SUB   #020H
        BR    APROV_COL

;*******TESTE
;ESC_VRAM:     BR    ESC_VRAM
;COD_MISC:     BR    COD_MISC
;***********
```

MISCELLA.ASM

```
COD_MISC:
        MOV   #LOW(TAB),ACC
        ST    TRL
        MOV   #HIGH(TAB),ACC
        ST    TRH
        LD    DEFINITIVA+1
        AND   #00FH
        ROL
        ADD   TRL
        ST    TRL
        AND   #000H             ;ZERA ACC
        ADDC  TRH
        ST    TRH
        CALL  PILHA
        bpco  MIS,1,IMPRIMI     ;VERIF. SE HOUVE CONT+CARACT
                                ; VAI P/ ESC_VRAM
        bro   XXX

PILHA:      PUSH  TRL
        PUSH  TRH
        RET


TAB:        jmpo   RCL_POP_ON
        jmpo   BSOLIDO
        jmpo   AOF
        jmpo   AOF
        jmpo   DER
        bro    RU_N
        bro    RU_N
        bro    RU_N
        bro    FON
        jmpo   RDC
        jmpo   TR
        jmpo   RTD
        jmpo   EDM
        jmpo   CR
        jmpo   ENM
        jmpo   EOC
        jmpo   TO_N
        jmpo   TO_N
        jmpo   TO_N


;Inicio ou fim do estilo Pop_on. Este reserva no máximo 4 linhas, ate receber um
;Resume Caption Loading (RCL). Este estilo mostra todas as linhas de uma
;vez.


RCL_POP_ON:
        ;SET1  MIS,0            ;(TESTE) HOUVE PRIMEIRO MISC
;       NOT1  OSDCR1,7          ;TORNA DISPLAY NAO VISIVEL
;****
;       MOV   #00CH,ROWR
```

```
;       MOV   #000H,CLMR
;       CLR1  OSDCR2,7
;       MOV   #064H,VRAM        ;100D
;       INC   CLMR
;       MOV   #000H,VRAM
;       INC   CLMR
;       MOV   #01100000B,VRAM
;       INC   CLMR
;       MOV   #10000000B,VRAM
;       INC   CLMR
;       SET1  OSDCR2,7
;       MOV   #10100000B,VRAM     ; BACKGROUND PRETO
;       INC   CLMR
;       MOV   #00001111B,VRAM     ; CARACTER BRANCO (DEFAULT)
;       LD    ROWR
;       ST    LINHA
;       MOV   #005,COLUNA
        RET


;Função BACK_SPACE

BSOLIDO:
        LD    COLUNA
        ST    CLMR
        bc    #004,SAI
        DEC   CLMR
        LD    CLMR
        ST    COLUNA
        CLR1  OSDCR2,7
        MOV   #020H,VRAM
SAI:    RET

;Sem função

AOF:    RET

;Coloca espaço em branco ate o final da linha.

DER:    LD    COLUNA
        ST    CLMR
        beo   #028H,XXX
        CLR1  OSDCR2,7
        MOV   #020H,VRAM
        INC   CLMR
        LD    CLMR
        ST    COLUNA
        RET

;Reserva 2-4 linhas para o estilo Roll_up

RU_N:
        RET


;Flash on

FON:    RET
```

```
;Resume Direct Caption - Inicio ou fim do estilo Paint_on

RDC:        RET

TR:         ; SET1   MIS,0
            RET

RTD:        SET1   MIS,0
            CLR1   OSDCR1,7
            MOV    #008,DECREMEN
            MOV    #004,ROWR
            MOV    #000,CLMR
            CLR1   OSDCR2,7

            MOV    #030H,ACC
COLUNA_0:   MOV    #00DH,VRAM
            ADD    VRAM
            ST     VRAM
            INC    ROWR
            DBNZ   DECREMEN,COLUNA_0
            MOV    #004H,ROWR
            MOV    #008,DECREMEN
            INC    CLMR
            MOV    #004H,ROWR
            MOV    #008,DECREMEN


COLUNA_1:   MOV    #000H,VRAM
            INC    ROWR
            DBNZ   DECREMEN,COLUNA_1
            INC    CLMR
            MOV    #004H,ROWR
            MOV    #008,DECREMEN


COLUNA_2:   MOV    #01110000B,VRAM
            INC    ROWR
            DBNZ   DECREMEN,COLUNA_2
            INC    CLMR
            MOV    #004H,ROWR
            MOV    #008,DECREMEN

COLUNA_3:   MOV    #10000000B,VRAM
            INC    ROWR
            DBNZ   DECREMEN,COLUNA_3
            INC    CLMR
            MOV    #004H,ROWR
            MOV    #008,DECREMEN

            SET1   OSDCR2,7
COLUNA_4:   MOV    #10100000B,VRAM          ;BACKGROUND BLACK
            INC    ROWR
            DBNZ   DECREMEN,COLUNA_4
            INC    CLMR
            MOV    #004H,ROWR
            MOV    #008,DECREMEN

COLUNA_5:   MOV    #00000111B,VRAM          ;CARACTER WHITE
```

```
        INC   ROWR
        DBNZ  DECREMEN,COLUNA_5
        SET1  OSDCR1,7
              MOV   #005H,COLUNA
        MOV   #005H,CLMR

        MOV   #004H,LINHA
        MOV   #004H,ROWR
        CLR1  OSDCR2,7
        RET


;Apaga a memória VRAM (esta' visível)

EDM:        NOT1  OSDCR1,7      ;torna não visível
        MOV   #00000000B,OSDCR2
        SET1  FECR1,7
        MOV   #000H,OSDCR1
        MOV   #027H,CLMR
        MOV   #00FH,ACC
L_1:      ST    ROWR
L_0:      MOV   #000H,VRAM
        DBNZ  CLMR,L_0
        MOV   #000H,VRAM
        MOV   #027H,CLMR
        DBNZ  ACC,L_1
        MOV   #027H,CLMR
        MOV   #000H,ROWR
L_2:      MOV   #000H,VRAM
        DBNZ  CLMR,L_2
        MOV   #000H,CLMR
        MOV   #000H,VRAM
        RET


CR:       SET1  MIS,7
        MOV   #006,COLUNA
        INC   LINHA
        LD    LINHA
        BNE   #00CH,NOVA_LINHA
        MOV   #00CH,LINHA
NOVA_LINHA:   RET


;Apaga a memória VRAM (esta' não visível)

ENM:        MOV   #027H,CLMR
        MOV   #00000000B,OSDCR2
        SET1  FECR1,7
        MOV   #000H,OSDCR1
        MOV   #027H,CLMR
        MOV   #00FH,ACC
L__1:     ST    ROWR
L__0:     MOV   #000H,VRAM
        DBNZ  CLMR,L__0
        MOV   #000H,VRAM
        MOV   #027H,CLMR
        DBNZ  ACC,L__1
        MOV   #027H,CLMR
```

```
        MOV    #000H,ROWR
L__2:     MOV    #000H,VRAM
        DBNZ   CLMR,L__2
        MOV    #000H,CLMR
        RET

;Fim da recepção do Pop-on (torna a memória VRAM visível)

EOC:      NOT1   OSDCR1,7
        RET

TO_N:     RET
```

CARACTER.ASM

```
CARACTERES:   MOV   #002H,DECREMEN      ;OS PARES SAO CARACTERES
        MOV   #AREA,R0          ;E ARMAZENA-OS EM DEFINITIVA
        MOV   #DEFINITIVA,R1
        LD    VILMA0
        ST    R2                ;CONTEUDO DE VILMA0 EM R2
CARACT:
        LD    @R0               ;CARREGA CPDH
        BN    PSW,0,BS
        LD    R2                ;VILMA0
        AND   #00001000B
        BNZ   BY_OK
        MOV   #000H,@R1
        BR    KKK

BY_OK:      LD    @R0
        CLR1  ACC,7
        ST    @R1
        BR    KKK

BS:       MOV   #07FH,@R1

KKK:        INC   R0
        INC   R1


        LD    @R0               ;CARREGA CPDL
        BN    PSW,0,BS2
        LD    R2                ;VILMA0
        AND   #00000100B
        BNZ   BYTE_OK2
        MOV   #000H,@R1
        BR    PULE_BYTE

BYTE_OK2:    LD    @R0
        CLR1  ACC,7
        ST    @R1
        BR    PULE_BYTE

BS2:      MOV   #07FH,@R1

PULE_BYTE:   INC   R0
        INC   R1


        LD    R2                ;O VALOR DE VILMA0 EH PERDIDO
        ROR
        ROR
        ROR
        ST    R2                ;VILMA0
        DBNZ  DECREMEN,CARACT
        BR    ESC_VRAM

;       BR    EXECUTA_CAR

;*******PARA TESTE***************
```

```
;EXECUTA_CAR:   BR     EXECUTA_CAR
;*******PARA TESTE***************

CARAC_CONT:
        MOV    #080H,AREA+2          ;OS PARES SAO
        MOV    #080H,AREA+3          ; CARACTERE+CONTROLE

        BR     CARACTERES
```

ESC_VRAM.ASM

```
ESC_VRAM:
            BN    MIS,0,FORA          ;SE MIS,0=1 CHEGOU MISC

        MOV   #00BH,BOTTOM_LINE     ;PARAMETRO DO MODO TEXTO

        BP    VILMA0,1,ICARO
        CLR1  OSDCR2,7
        BR    BARROS

ICARO:      SET1  OSDCR2,7          ;PREPARA PARA ATRIBUTO

BARROS:     CALL  DISLAY
        LD    DEFINITIVA          ;COLOCA NA VRAM O 1o BYTE
        BZ    P_2               ;(CARACTERE OU CONTROLE)
              ST      VRAM
              INC     COLUNA

P_2:        CLR1  OSDCR2,7
        CALL  DISLAY
        LD    DEFINITIVA+1        ;COLOCA NA VRAM 2o BYTE
        BZ    P_3
        ST    VRAM              ; (CARACTERE)
        INC   COLUNA

IMPRIMI:
        CALL  DISLAY
P_3:      LD    DEFINITIVA+2        ;COLOCA NA VRAM 3o BYTE
        BZ    P_4
        ST    VRAM              ; (CARACTERE)
        INC   COLUNA

        CALL  DISLAY
P_4:      LD    DEFINITIVA+3        ;COLOCA NA VRAM 4o BYTE
        bzo   XXX
        ST    VRAM              ; (CARACTERE)
        INC   COLUNA
FORA:       JMP   XXX


;*****************
;Rotina para identificar a linha e coluna
;testando se clmr>027H e rowr>0FH
;*****************


DISLAY:     LD    LINHA
;****DISLAY

        LD    LINHA
        SUB   BOTTOM_LINE
              BP      PSW,7,LINE_OK
              LD      BOTTOM_LINE
              ST      ROWR
              BR      COL
LINE_OK:      LD      LINHA
```

```
            ST      ROWR
COL:        LD      COLUNA
            SUB     #027H
            BP      PSW,7,COL_OK
            MOV     #027H,CLMR
            MOV     #027H,COLUNA
            BR      CONTINUA
COL_OK:             LD      COLUNA
            ST      CLMR

CONTINUA:   RET
```

PREAMBLE ADDRESS CODES

	L. 1	L. 2	L. 3	L. 4	L. 5	L. 6	L. 7	L. 8	L. 9	L. 10	L. 11	L. 12	L. 13	L. 14	L. 15
1 byte do par:															
Canal de dados 1	11	11	12	12	15	15	16	16	17	17	10	13	13	14	14
Canal de dados 2	19	19	1A	1A	1D	1D	1E	1E	1F	1F	18	18	1B	1C	1C
2 byte do par:															
Branco	40	60	40	60	40	60	40	60	40	60	40	40	60	40	60
Branco sublinhado	41	61	41	61	41	61	41	61	41	61	41	41	61	41	61
Verde	42	62	42	62	42	62	42	62	42	62	42	42	62	42	62
Verde sublinhado	43	63	43	63	43	63	43	63	43	63	43	43	63	43	63
Azul	44	64	44	64	44	64	44	64	44	64	44	44	64	44	64
Azul sublinhado	45	65	45	65	45	65	45	65	45	65	45	45	65	45	65
Ciano	46	66	46	66	46	66	46	66	46	66	46	46	66	46	66
Ciano sublinhado	47	67	47	67	47	67	47	67	47	67	47	47	67	47	67
Vermelho	48	68	48	68	48	68	48	68	48	68	48	48	68	48	68
Vermelho sublinhado	49	69	49	69	49	69	49	69	49	69	49	49	69	49	69
Amarelo	4A	6A	4A	6A	4A	6A	4A	6A	4A	6A	4A	4A	6A	4A	6A
Amarelo sublinhado	4B	6B	4B	6B	4B	6B	4B	6B	4B	6B	4B	4B	6B	4B	6B
Magenta	4C	6C	4C	6C	4C	6C	4C	6C	4C	6C	4C	4C	6C	4C	6C
Magenta sublinhado	4D	6D	4D	6D	4D	6D	4D	6D	4D	6D	4D	4D	6D	4D	6D
Branco itálico	4E	6E	4E	6E	4E	6E	4E	6E	4E	6E	4E	4E	6E	4E	6E
Branco itálico sublinhado	4F	6F	4F	6F	4F	6F	4F	6F	4F	6F	4F	4F	6F	4F	6F
Identação 0	50	70	50	70	50	70	50	70	50	70	50	50	70	50	70
Identação 0 sublinhado	51	71	51	71	51	71	51	71	51	71	51	51	71	51	71
Identação 4	52	72	52	72	52	72	52	72	52	72	52	52	72	52	72
Identação 4 sublinhado	53	73	53	73	53	73	53	73	53	73	53	53	73	53	73
Identação 8	54	74	54	74	54	74	54	74	54	74	54	54	74	54	74
Identação 8 sublinhado	55	75	55	75	55	75	55	75	55	75	55	55	75	55	75
Identação 12	56	76	56	76	56	76	56	76	56	76	56	56	76	56	76
Identação 12 sublinhado	57	77	57	77	57	77	57	77	57	77	57	57	77	57	77
Identação 16	58	78	58	78	58	78	58	78	58	78	58	58	78	58	78
Identação 16 sublinhado	59	79	59	79	59	79	59	79	59	79	59	59	79	59	79
Identação 20	5A	7A	5A	7A	5A	7A	5A	7A	5A	7A	5A	5A	7A	5A	7A
Identação 20 sublinhado	5B	7B	5B	7B	5B	7B	5B	7B	5B	7B	5B	5B	7B	5B	7B
Identação 24	5C	7C	5C	7C	5C	7C	5C	7C	5C	7C	5C	5C	7C	5C	7C
Identação 24 sublinhado	5D	7D	5D	7D	5D	7D	5D	7D	5D	7D	5D	5D	7D	5D	7D
Identação 28	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E	5E
Identação 28 sublinhado	5F	7F	5F	7F	5F	7F	5F	7F	5F	7F	5F	5F	7F	5F	7F

MID ROW CODES

Canal de dados 1	Canal de dados 2	Atributo
11 20	19 20	branco
11 21	19 21	branco sublinhado
11 22	19 22	verde
11 23	19 23	verde sublinhado
11 24	19 24	azul
11 25	19 25	azul sublinhado
11 26	19 26	ciano
11 27	19 27	ciano sublinhado
11 28	19 28	vermelho
11 29	19 29	vermelho sublinhado
11 2A	19 2A	amarelo
11 2B	19 2B	amarelo sublinhado
11 2C	19 2C	magenta
11 2D	19 2D	magenta sublinhado
11 2E	19 2E	itálico
11 2F	19 2F	itálico sublinhado

MISCELLANEOUS CONTROL CODES

Canal de dados 1	Canal de dados 2	Mneumônico	Descrição do comando
14 20	1C 20	RCL	*resume caption loading*
14 21	1C 21	BS	*backspace*
14 22	1C 22	AOF	*reservado*
14 23	1C 23	AON	*reservado*
14 24	1C 24	DER	*deleto to end of row*
14 25	1C 25	RU2	*row-up captions - 2*
14 26	1C 26	RU3	*row-up captions - 3*
14 27	1C 27	RU4	*row-up captions - 4*
14 28	1C 28	FON	*flash O*
14 29	1C 29	RDC	*resume direct captioning*
14 2A	1C 2A	TR	*text restart*
14 2B	1C 2B	RTD	*resume text display*
14 2C	1C 2C	EDM	*erase displayed memory*
14 2D	1C 2D	CR	*carriage return*
14 2E	1C 2E	ENM	*erase non-displayed memory*
14 2F	1C 2F	EOC	*end of caption*
17 21	1F 21	TO1	*tab offset column 1*
17 22	1F 22	TO2	*tab offset column 2*
17 23	1F 23	TO3	*tab offset column 3*

					ROTINAS *.ASM	
	LIM_VRAM	**TESTE**	**CAPTURA**	**ANÁLISE**	**CONT_CAR**	**CONTROLE**
ENTRADAS		ROWR=0	MIS=0	ACC=AREA	ACC=AREA	DECREMEN=2
		CLMR=0	R0=#AREA			R0=#AREA+1
			DECREMEN=2			R1=#AREA
			INCREMEN=0			
SAÍDAS	ROWR=0	MIS=0	INCREMEN=0	VILMA0	DEFINITIVA=AREA	VILMA1
	CLMR=0	R0=#AREA	R0=#AREA	ACC=000000XX	DEFINITIVA+1=AREA1	DEFINITIVA=AREA(N)
		DECREMEN=2	ACC=2	DECREMEN=0	DEFINITIVA+2=AREA2	DEFINITIVA+1=AREA(N+!)
		INCREMEN=0		AREA=CPDH 1	DEFINITIVA+3=AREA3	DEFINITIVA+2=0
				AREA1=CPDL1		DEFINITIVA+3=0
				AREA2=CPDH2		**DECREMEN=1**
				AREA3=CPDL2		

EXEC_CON	MISCELLA	ESC_VRAM	CARACTER	CARAC_CONT	EXEC_CAR
DEFINITVA,7=0		ACC=LINHA	DECREMEN=002H		MIS,0
DEFINITIVA+1,7=0			R0= #AREA		BOTTOM_LINE
			R1= #DEFINITIVA		DECREME=4
					R1=#DEFINITIVA
					LINHA
					COLUNA
SAÍDA MRC	**SAÍDA MODO TEXTO (RTD)**	ROWR=LINHA	DECREMEN=0	DEFINITIVA=CPDH1	
DEFINITIVA=TAB_MRC+OFFSET	CLMR=5	CLMR=COLUNA	VILMA0	DEFINITIVA+1=0	
DEFINITIVA+1=0	COLUNA=5			DEFINITIVA+2=0	
SAÍDA PAC	LINHA=4			DEFINITIVA+2=0	
DEFINITIVA=TAB_MRC+OFFSET	ROW=4				
DEFINITIVA+1=0	OSDCR2,7=1				
LINHA = TAB_LINHA+OFFSET	MIS,0=1				
SAÍDA IDENTAÇÃO	**SAÍDA RET CARRO (CR)**				
LINHA=TAB_LINHA+OFFSET	COLUNA=6				
CLMR=COLUNA+N (PAR_IND)	LINHA=LINHA+1				
COLUNA=CLMR	**OU**				
VRAM(N)=020H	LINHA=4				

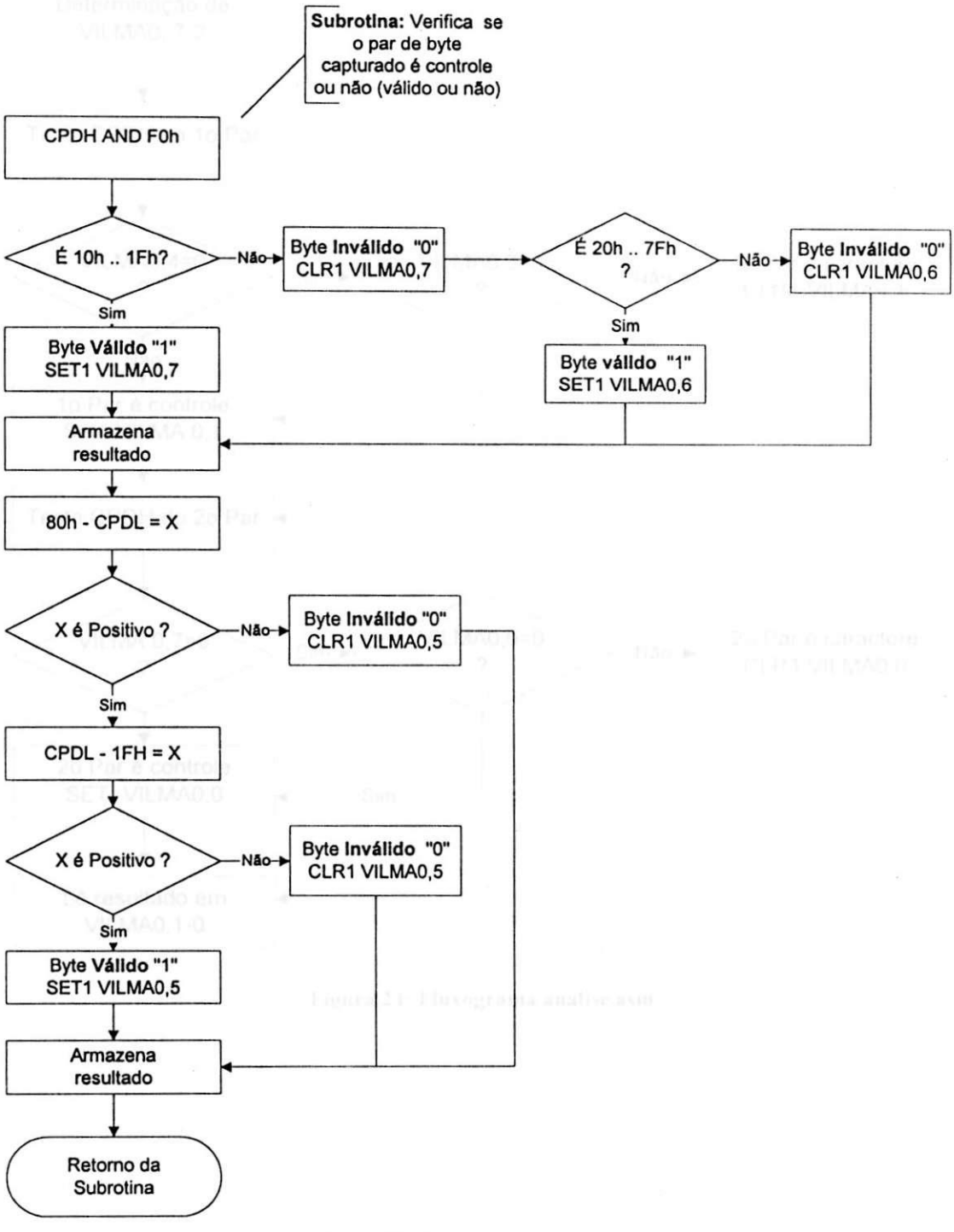


Figura 23: Fluxograma analise.asm

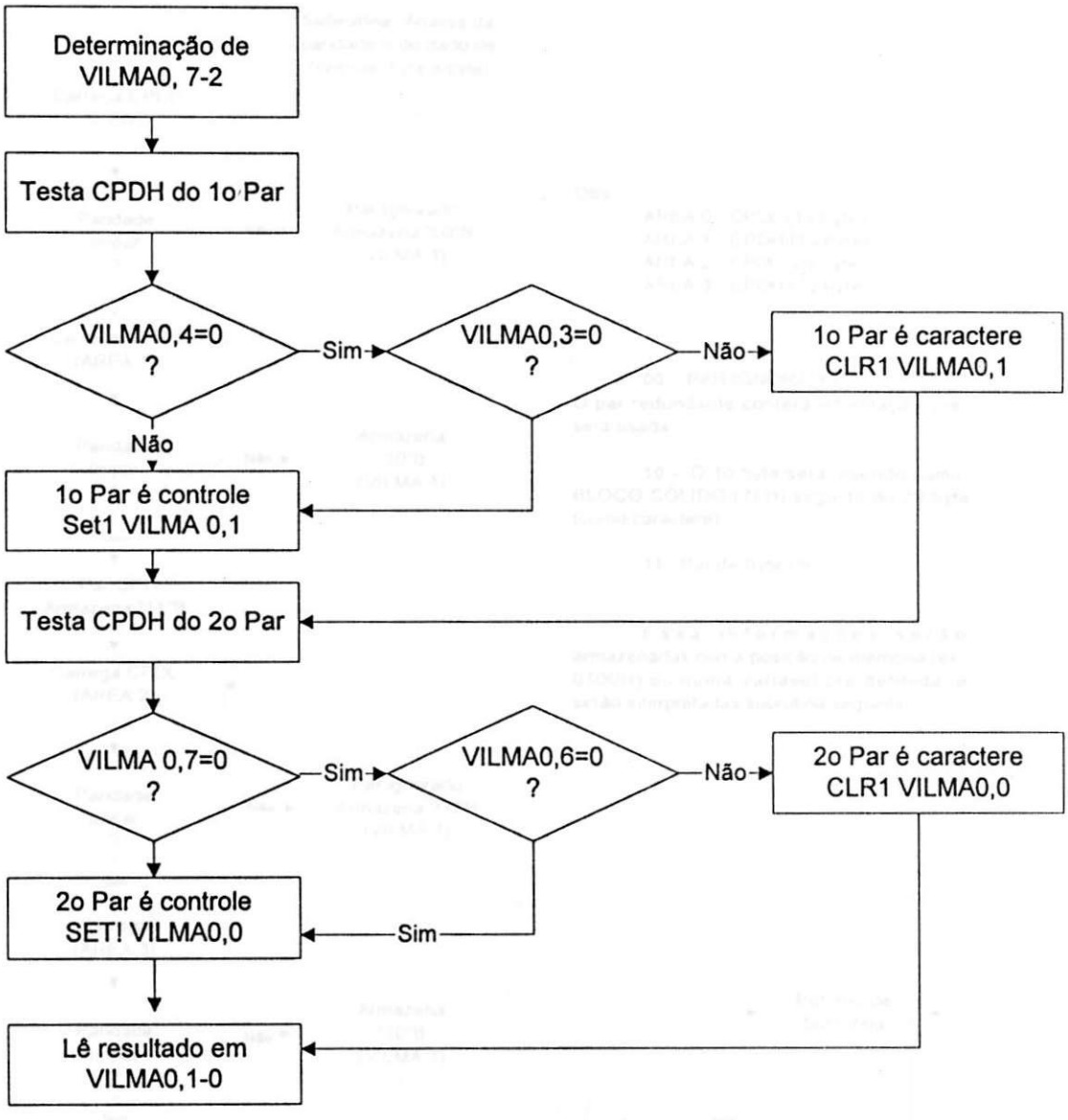


Figura 24: Fluxograma analise.asm

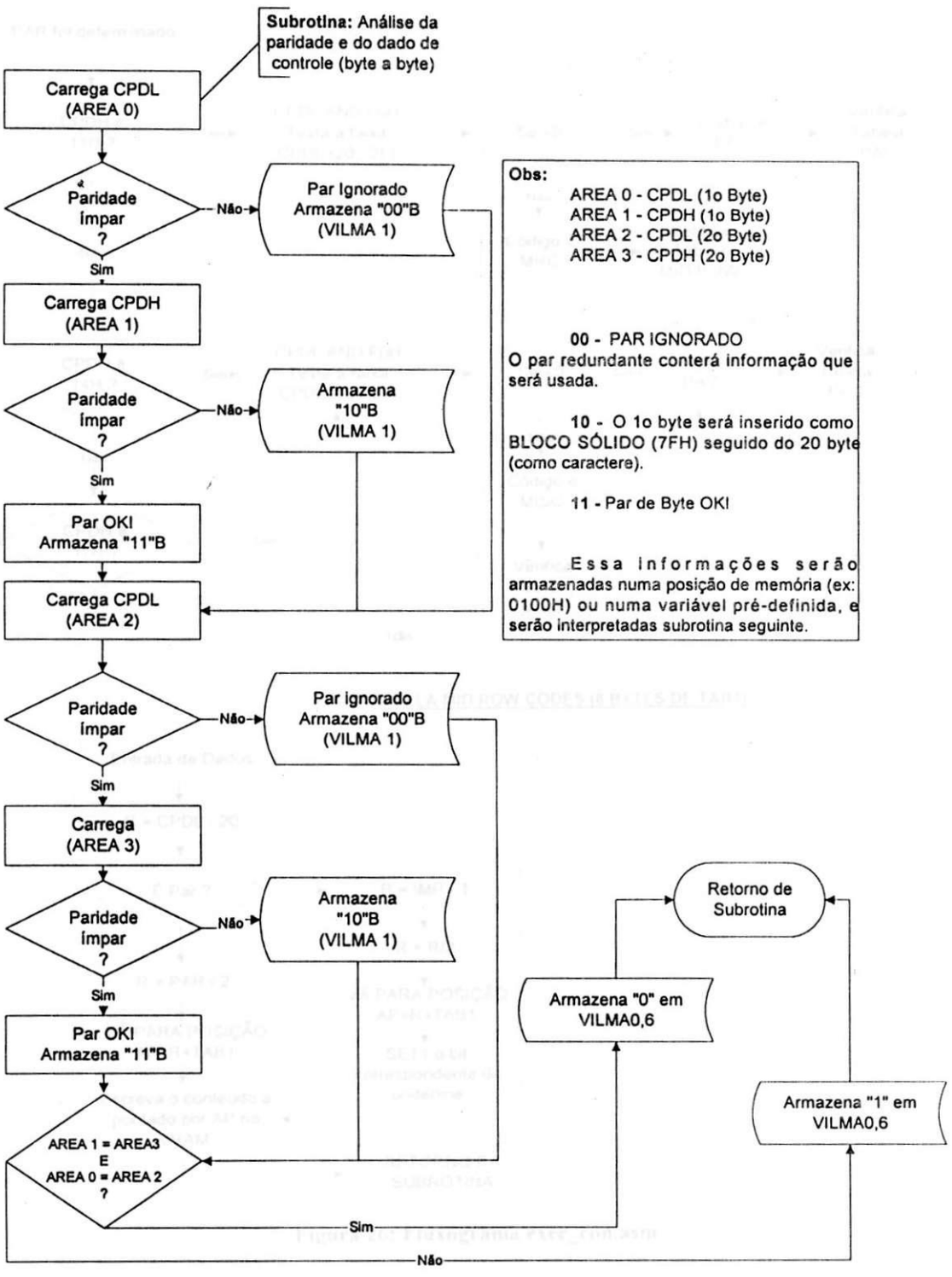
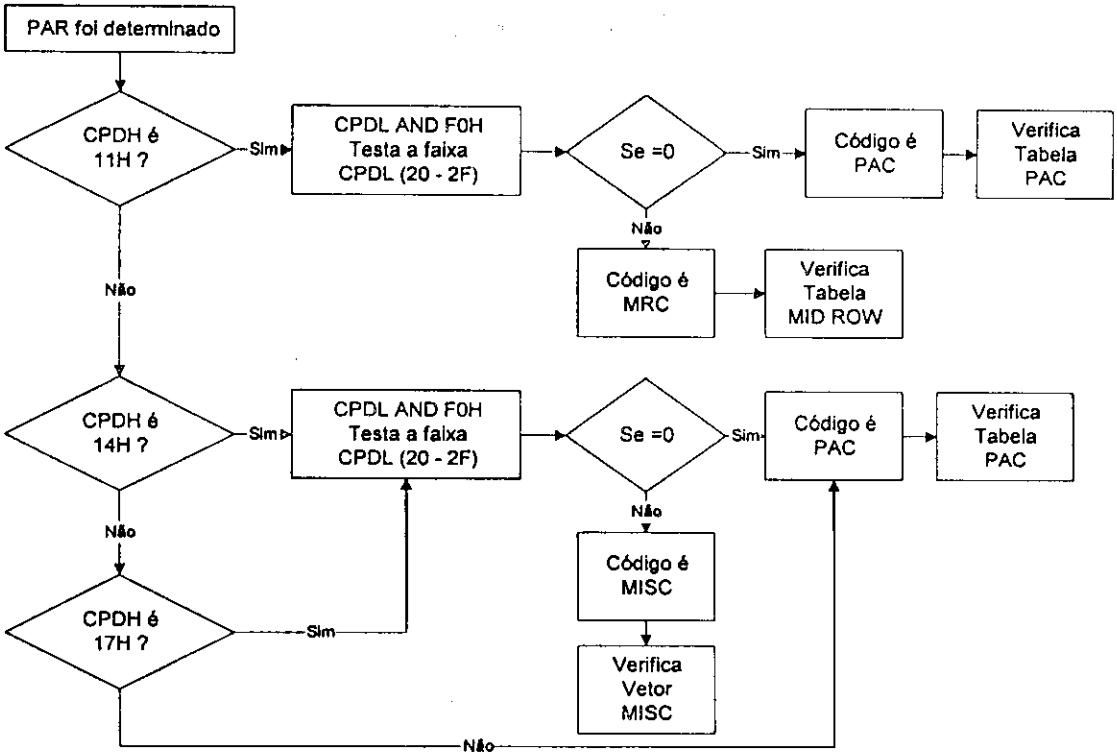


Figura 25: Fluxograma controle.asm



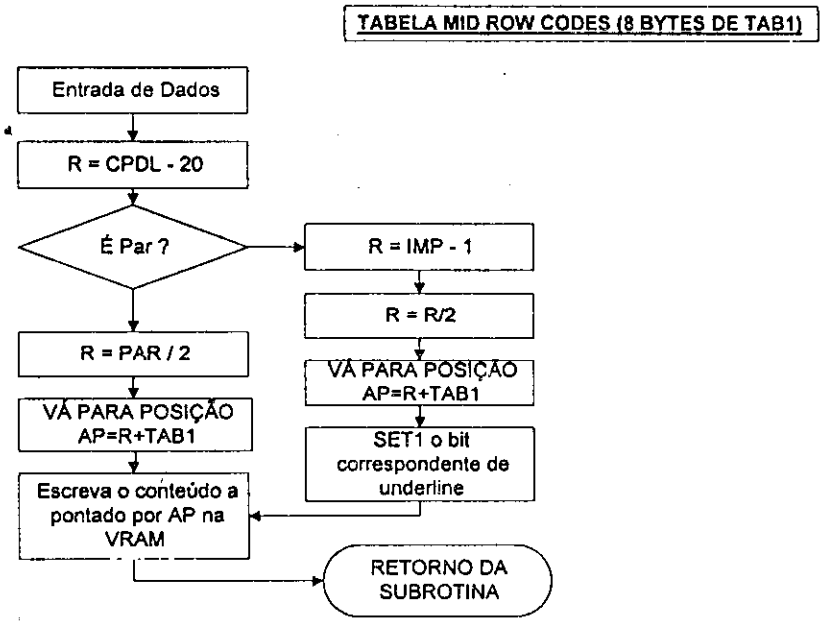


Figura 26: Fluxograma exec_con.asm