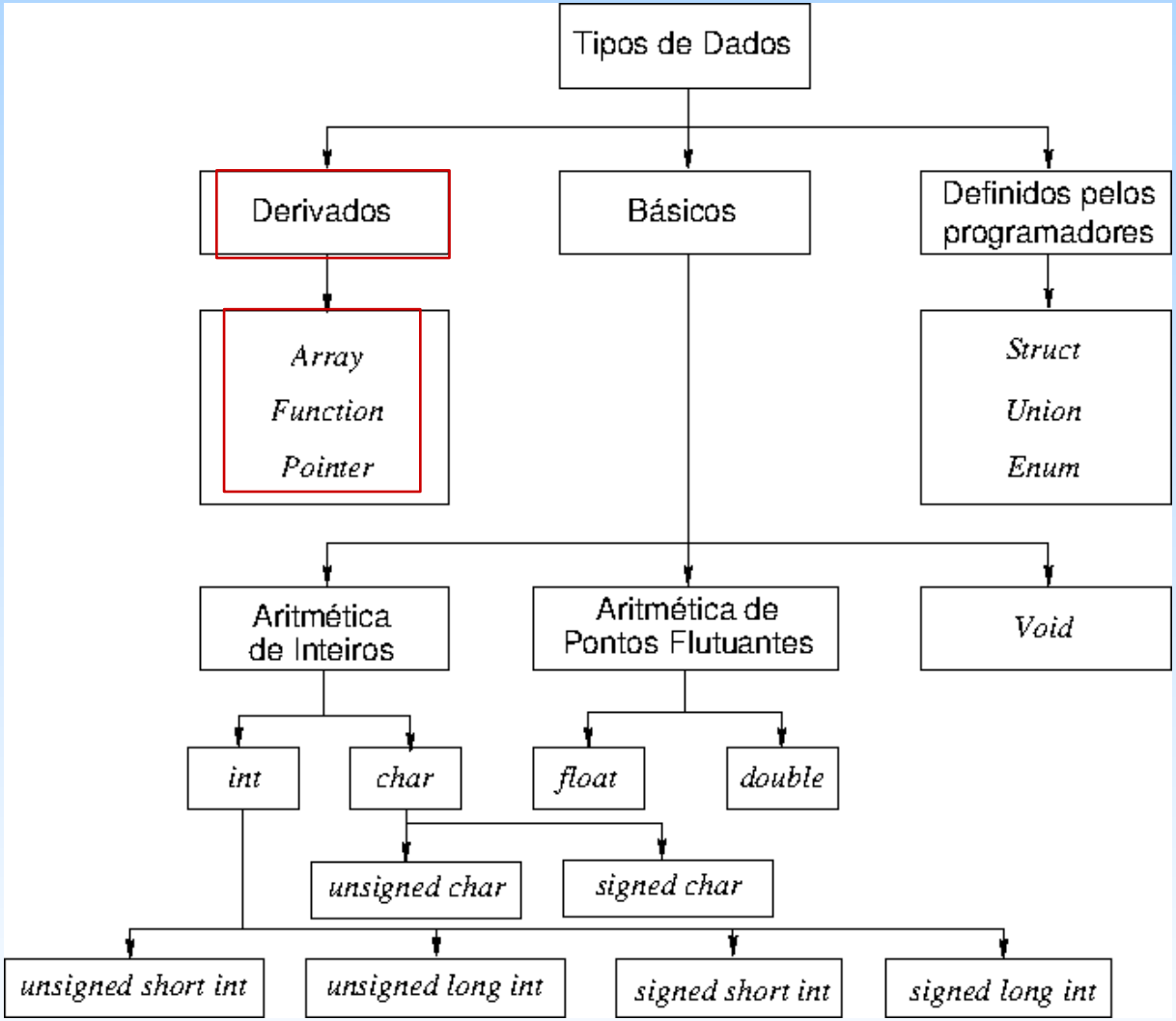
EA871

LCD

Strings em C

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Tipos de Dados Derivados



Strings: **arranjos** (*arrays*) de elementos de tipo char (*um byte*).

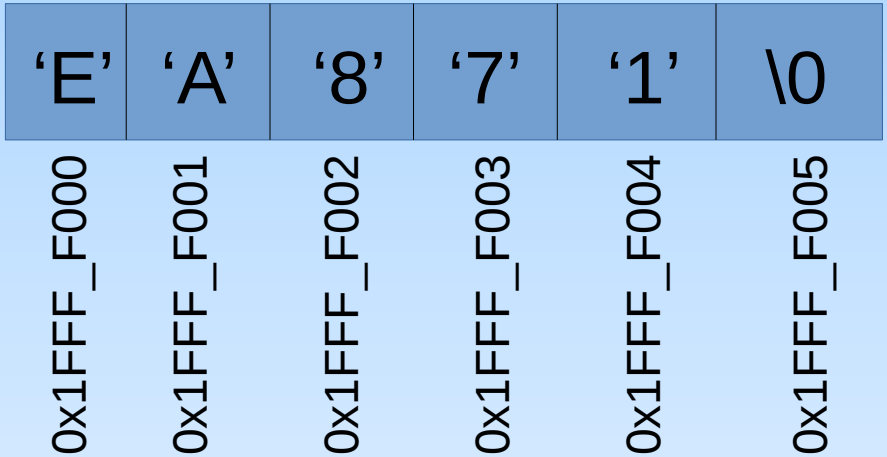
Declarações de *strings* de n caracteres

- Declaração de um arranjo com n elementos de tipo char, **sem** inicialização. Por exemplo, n=6:
  - char stringD[6];
- Declaração explícita de um arranjo com n elementos de tipo char , **com** inicilização. Por exemplo, n=6:
  - char stringD[6] = {‘E’, ‘A’, ‘8’, ‘7’, ‘1’,’\0’};
  - **Em C, convenciona-se usar o caractere NULL ‘\0’ para terminar um arranjo de caracteres.**
- Declaração implícita de um arranjo do tipo char, **com** inicialização
  - char stringD[] = “EA871”;
  - **O compilador aloca automaticamente um elemento a mais para acomodar o caractere NULL ‘\0**’.

Definição de *strings* de n caracteres

- Alocação de espaços de memória para *strings* declaradas

char stringD[] = “EA871”;



- Endereço inicial de uma *string*
  - char *end_stringD;
  - end_stringD = stringD = &stringD[0] = (char *)0x1FFFF000;
- Acesso a n-ésimo caractere de uma *string* (p.ex., n=3)
  - char caractere;
  - caractere = stringD[3] = *(stringD+3) = *((char *)0x1FFFF003);

Códigos Binários dos Caracteres

- **ASCII (*American Standard Code for Information Interchange*)**: Código representável em 1 *byte*
  - Códigos de 7 *bits*

- UTF (*Unicode transformation Format*)-8: código *multibyte*
  - *1 byte: ASCII*
  - *2 bytes: caracteres latinos e do Oriente Médio*
  - *3 bytes: ideogramas chineses, alfabetos japoneses e coreanos, etc.*
  - *4 bytes: símbolos matemáticos, emoji, etc.*

Códigos ASCII

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	&#032;	Space	64	40	100	&#064;	@	96	60	140	&#096;	`
1	1	001	SoH	33	21	041	&#033;	!	65	41	101	&#065;	A	97	61	141	&#097;	a
2	2	002	SoTxt	34	22	042	&#034;	"	66	42	102	&#066;	B	98	62	142	&#098;	b
3	3	003	EoTxt	35	23	043	&#035;	#	67	43	103	&#067;	C	99	63	143	&#099;	c
4	4	004	EoT	36	24	044	&#036;	$	68	44	104	&#068;	D	100	64	144	&#100;	d
5	5	005	Enq	37	25	045	&#037;	%	69	45	105	&#069;	E	101	65	145	&#101;	e
6	6	006	Ack	38	26	046	&#038;	&	70	46	106	&#070;	F	102	66	146	&#102;	f
7	7	007	Bell	39	27	047	&#039;	'	71	47	107	&#071;	G	103	67	147	&#103;	g
8	8	010	Bsp	40	28	050	&#040;	(	72	48	110	&#072;	H	104	68	150	&#104;	h
9	9	011	HTab	41	29	051	&#041;	)	73	49	111	&#073;	I	105	69	151	&#105;	i
10	A	012	LFeed	42	2A	052	&#042;	*	74	4A	112	&#074;	J	106	6A	152	&#106;	j
11	B	013	VTab	43	2B	053	&#043;	+	75	4B	113	&#075;	K	107	6B	153	&#107;	k
12	C	014	FFeed	44	2C	054	&#044;	,	76	4C	114	&#076;	L	108	6C	154	&#108;	l
13	D	015	CR	45	2D	055	&#045;	-	77	4D	115	&#077;	M	109	6D	155	&#109;	m
14	E	016	SOut	46	2E	056	&#046;	.	78	4E	116	&#078;	N	110	6E	156	&#110;	n
15	F	017	SIn	47	2F	057	&#047;	/	79	4F	117	&#079;	O	111	6F	157	&#111;	o
16	10	020	DLE	48	30	060	&#048;	0	80	50	120	&#080;	P	112	70	160	&#112;	p
17	11	021	DC1	49	31	061	&#049;	1	81	51	121	&#081;	Q	113	71	161	&#113;	q
18	12	022	DC2	50	32	062	&#050;	2	82	52	122	&#082;	R	114	72	162	&#114;	r
19	13	023	DC3	51	33	063	&#051;	3	83	53	123	&#083;	S	115	73	163	&#115;	s
20	14	024	DC4	52	34	064	&#052;	4	84	54	124	&#084;	T	116	74	164	&#116;	t
21	15	025	NAck	53	35	065	&#053;	5	85	55	125	&#085;	U	117	75	165	&#117;	u
22	16	026	Syn	54	36	066	&#054;	6	86	56	126	&#086;	V	118	76	166	&#118;	v
23	17	027	EoTB	55	37	067	&#055;	7	87	57	127	&#087;	W	119	77	167	&#119;	w
24	18	030	Can	56	38	070	&#056;	8	88	58	130	&#088;	X	120	78	170	&#120;	x
25	19	031	EoM	57	39	071	&#057;	9	89	59	131	&#089;	Y	121	79	171	&#121;	y
26	1A	032	Sub	58	3A	072	&#058;	:	90	5A	132	&#090;	Z	122	7A	172	&#122;	z
27	1B	033	Esc	59	3B	073	&#059;	;	91	5B	133	&#091;	[	123	7B	173	&#123;	{
28	1C	034	FSep	60	3C	074	&#060;	<	92	5C	134	&#092;	\	124	7C	174	&#124;	|
29	1D	035	GSep	61	3D	075	&#061;	=	93	5D	135	&#093;	]	125	7D	175	&#125;	}
30	1E	036	RSep	62	3E	076	&#062;	>	94	5E	136	&#094;	^	126	7E	176	&#126;	~
31	1F	037	USep	63	3F	077	&#063;	?	95	5F	137	&#095;	_	127	7F	177	&#127;	Delete

charstable.com

São aplicáveis operações lógico-aritméticas sobre os elementos de uma *string*.

Operações sobre *strings*

- Códigos ASCII e UTF-8 amplamente usados em processadores de textos, páginas da *web* e mensagens na internet.
- Funções disponíveis na biblioteca-padrão de C, dedicadas para manipulação de arranjos de caracteres em ASCII (*strings*)
  - *Strings* terminadas com o caractere NULL, ‘\0’
  - Inclusão do arquivo-cabeçalho contendo todas as declarações das funções relacionadas às operações sobre *strings*
    - string.h

Algumas funções em C

Int strcmp (char *str1, char *str2)	Compara as *strings* str1 e str2. Retorna 0 (str1==str2), <0 (str1 < str2) e >0 (str1 > str2)
Int strncmp (char *str1, char *str2, size_t n)	Compara os n primeiros caracteres das *strings* str1 e str2.
char *strcpy (char * dest, char * src)	Copia *string* src no espaço alocado para *string* dest.
char *strncpy (char *dest, char *src, size_t n)	Copia n primeiros caracteres da *string* src no espaço alocado para *string* dest.
unsigned int strlen (char *str)	Retorna a quantidade de caracteres na *string* str, sem contar o caractere ‘\0’.
char *strchr (char *str, int c)	Retorna o endereço do primeiro elemento da *string* str que contém o caractere c.
char * strcat (char *dest, char *src)	Acrescenta string src no fim da *string* dest sobrescrevendo o caractere ‘\0’ de dest.
char *strncat (char * dest, char *src, size_t n)	Concatena n primeiros caracteres da *string* src no fim da *string* dest, sobreescrevendo o caractere ‘\0’.
char *strtok (char *str, const char *sep)	Extrai recorrentemente os tokens da *string* str entre os separadores sep

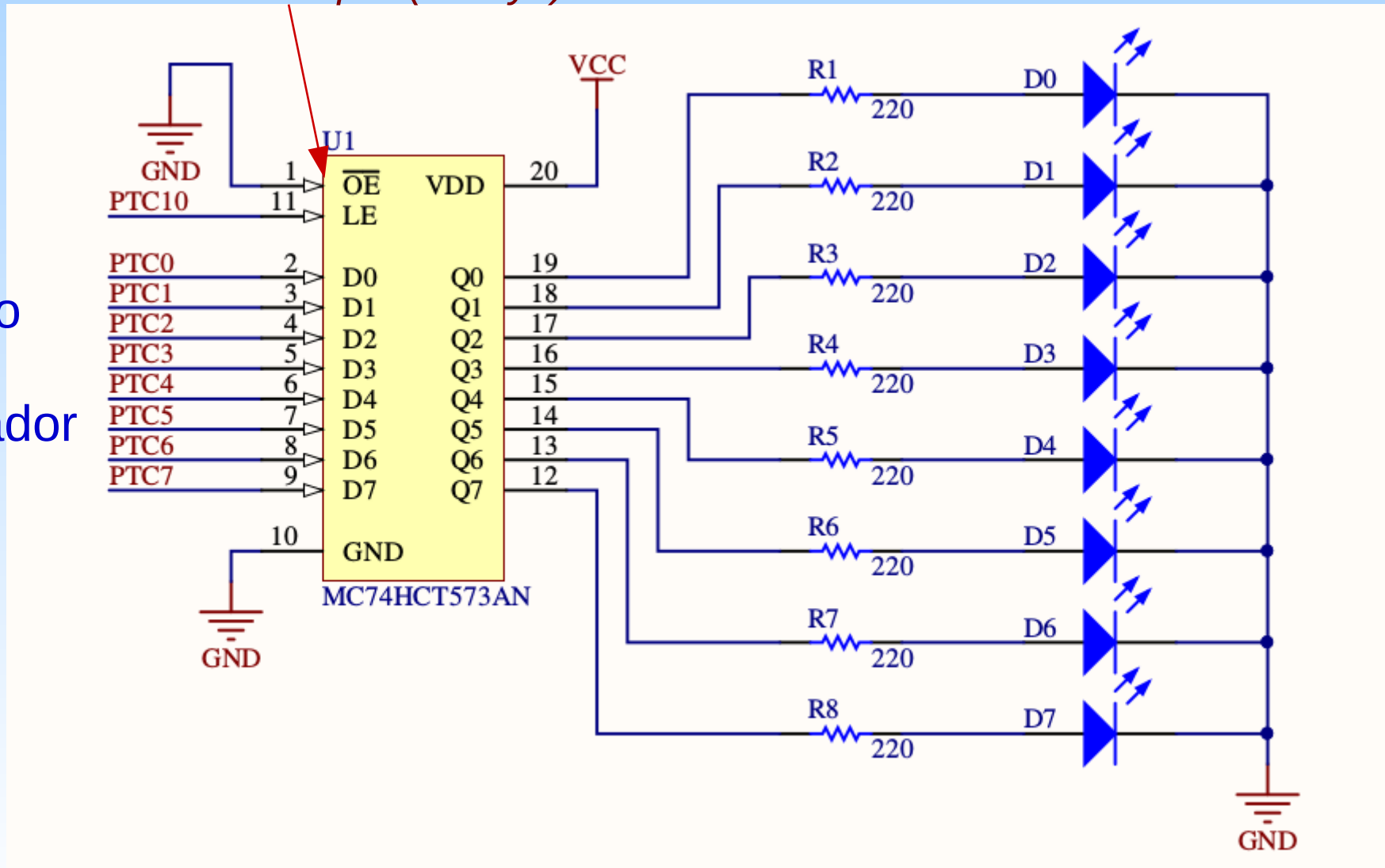
Projeto-exemplo

- Visualização dos códigos ASCII dos caracteres de uma *string* via *leds* vermelhos do *shield* FEEC871.
- A *string* é resultado de diferentes operações sobre vetores cujos elementos são do tipo char.

Hardware

/OE = 0 → *Output (always) enabled*

Pinos do micro controlador



Microcontrolador – *Latch* 74573

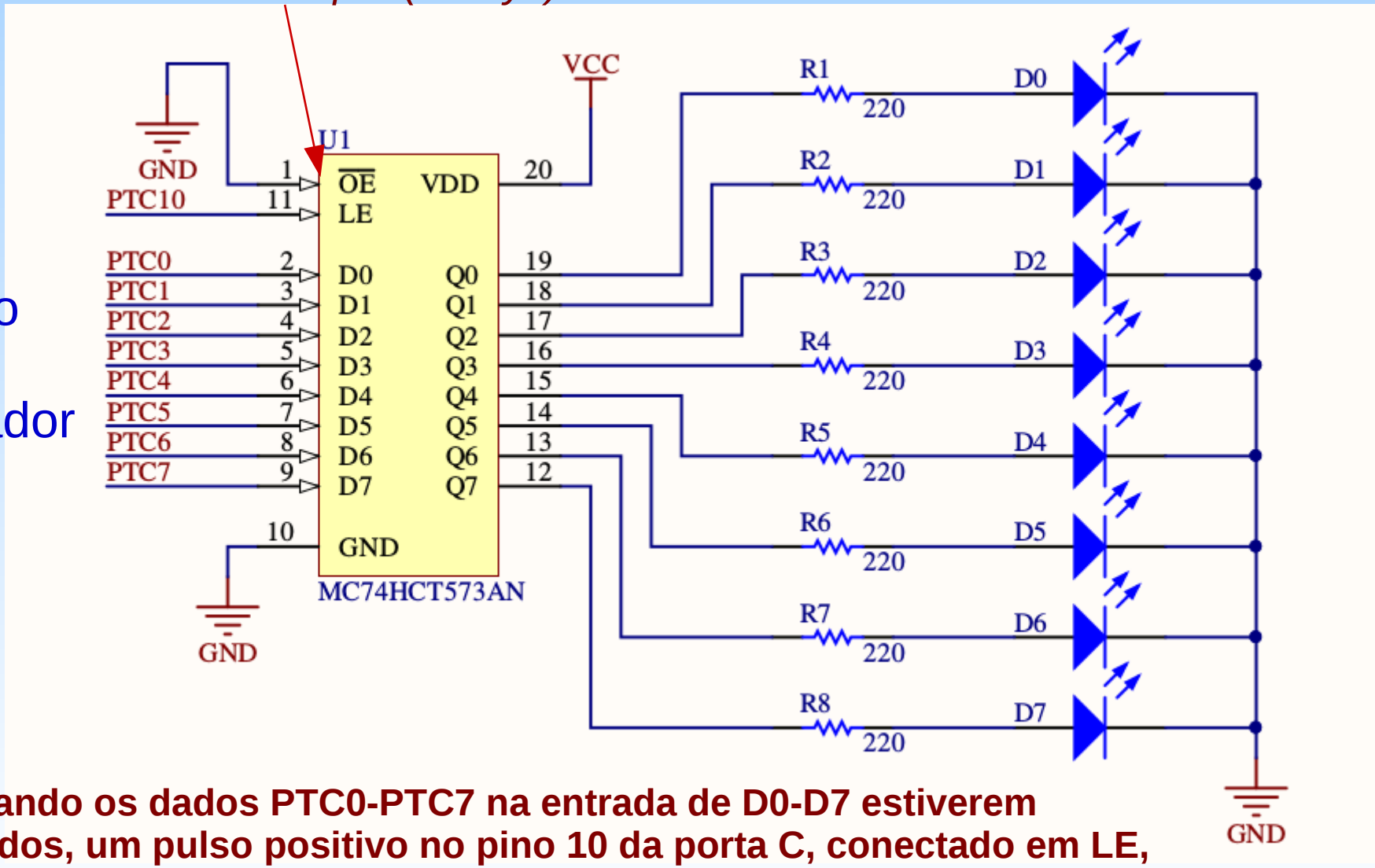
	Microcontrolador	*Latch*
Funcional	1/0	Habilitado/Desabilitado (LE)
Elétrico	3V3/0V	3.4V (min. 2.7V)/0.35V (min. -)
Temporal	1 ciclo de instrução ~50ns	Mínima largura do tempo de LE: 15ns; *setup*: 7ns;*hold*: 10ns
Mecânico	9 pinos da porta C (Terra comum)	8 pinos de dados Di e 1 pino de controle LE (Terra comum)

Datasheet de 74573: https://datasheetspdf.com/pdf-file/629141/national/74573/1

Hardware

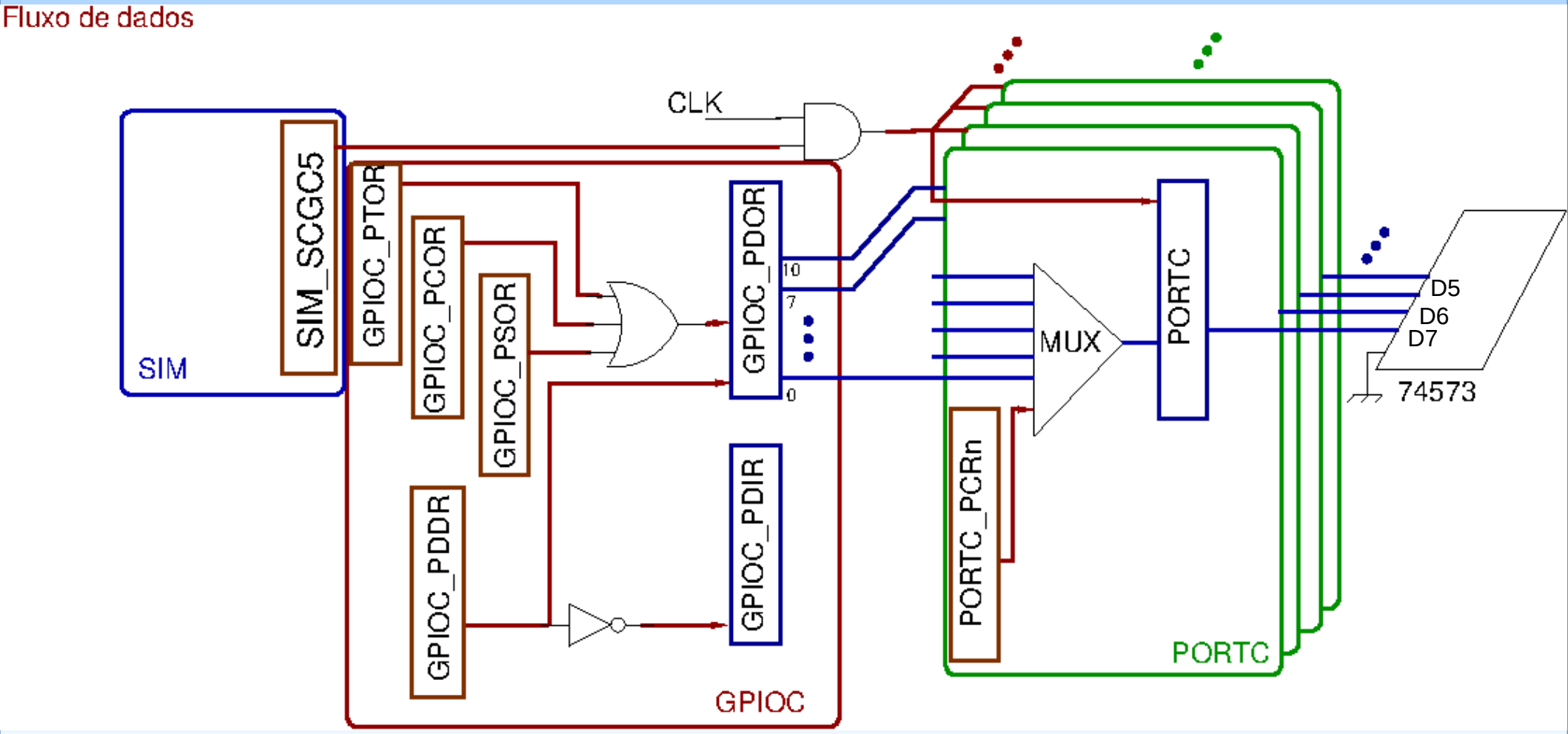
/OE = 0 → *Output (always) enabled*

Pinos do micro controlador



Quando os dados PTC0-PTC7 na entrada de D0-D7 estiverem estabilizados, um pulso positivo no pino 10 da porta C, conectado em LE, habilitará o *latch* para reter os valores DO-D7 nele.

Fluxo de Dados



Registradores

- **SIM_SCGC5**
  - Setar *bit* 11 para habilitar o *clock* da porta C
- **PORTC_PCR0 a PORTC_PCR7 e PORTC_PCR10**
  - Setar *bits* 8, 9 e 10 para definir o sinal digital nos pinos 0-7,10
- **GPIOC_PDDR**
  - Sentido dos sinais nos *bits* 0-7, 10: saída
- **GPIOC_PDOR/GPIOC_PSOR/GPIOC_PCOR**
  - *bit* 10: LE
  - *bit* 0-7: D0-D7

Técnicas de Programação

- É possível escrever **paralelamente** um *byte* (8 *bits*) nos 8 pinos D0-D7 do *latch* 74573 com um número mínimo de comandos em C?
  - Os 8 *bits* correspondem ao *byte* menos significativo do registrador GPIOC_PDOR (32 *bits*)
  - Conversão explícita de uma variável do tipo char (8 *bits*) para tipo unsigned int (32 *bits*) como os *bits* mais significativos resetados.
  - Duas alternativas:
    - Operações lógicas *bit* a *bit* no registrador GPIOC_PDOR
    - Operações setar e limpar os *bits* do registrador GPIOC_PDOR através dos registradores de controle GPIOC_PSOR e GPIOC_PCOR.

Técnicas de Programação

- Alternativa 1:
  - Limpar o *byte* menos significativo:
    - **GPIOC_PDOR** &= 0xFFFFFF00;
  - Setar os bits em 1 no *byte* menos significativo:
    - **GPIOC_PDOR** |= **(unsigned int) c**;
- Alternativa 2:
  - Limpar o *byte* menos significativo escrevendo 1 em GPIOC_PCOR:
    - **GPIOC_PCOR** = 0x000000FF;
  - Escrever os *bits* 1 de c em GPIOC_PSOR:
    - **GPIOC_PSOR** = **(unsigned int) c**;

Técnicas de Programação

- Modularização de códigos
  - Inserção de atraso de 0.5ms para ver o código exibido nos *leds* R:

    delay (unsigned int t)
  - Inicialização do controle dos pinos D0-D7 e LE do *latch* 74573: GPIO_initLatch74573()
  - Atribuição de valores aos pinos DO-D7 do *latch* :

    GPIO_setByte (char c)
  - Habilitação do *latch*: gerar por *software* um pulso de habilitação de ~50ns

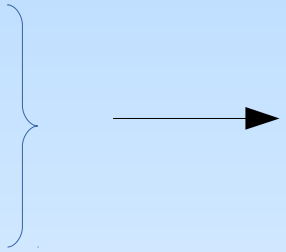
    GPIO_enableLE ();

Pseudocódigo

GPIO_initLatch74573();

Laço para mostrar os caracteres de uma string:

- i = 0;
- Enquanto (str[i] != '\0')
  - GPIO_setData (str[i]);
  - GPIO_enableLE();
  - delay2us (500000);
  - i++;



escreveLatch (str[i]);

Dígitos: 0x30-0x39

Maiúsculas: 0x41-0x5A

Minúsculas: 0x61-0x7A

CodeWarrior IDE Development Suite

Informações Adicionais

- Amostra simplificada da interface da biblioteca string

  https://www.ime.usp.br/~pf/algoritmos/apend/string.h.html
- Linguagem C: Representação de Dados

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/apostila_C/RepresentacaoDados.pdf
- *Datasheet 74573*

  https://datasheetspdf.com/pdf-file/629141/national/74573/1
- *Esquemático do shield FEEC871*

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/Esquematico_EA871-Rev3.pdf
- *KL25 Sub-Family Reference Manual*

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
  - Mapa de memória: Seção 4.2 (página 105)
  - SIM: Capítulo 12 (página 192)
  - PORT: Capítulo 11 (página 175)
  - GPIO: Capítulo 41 (página 771)

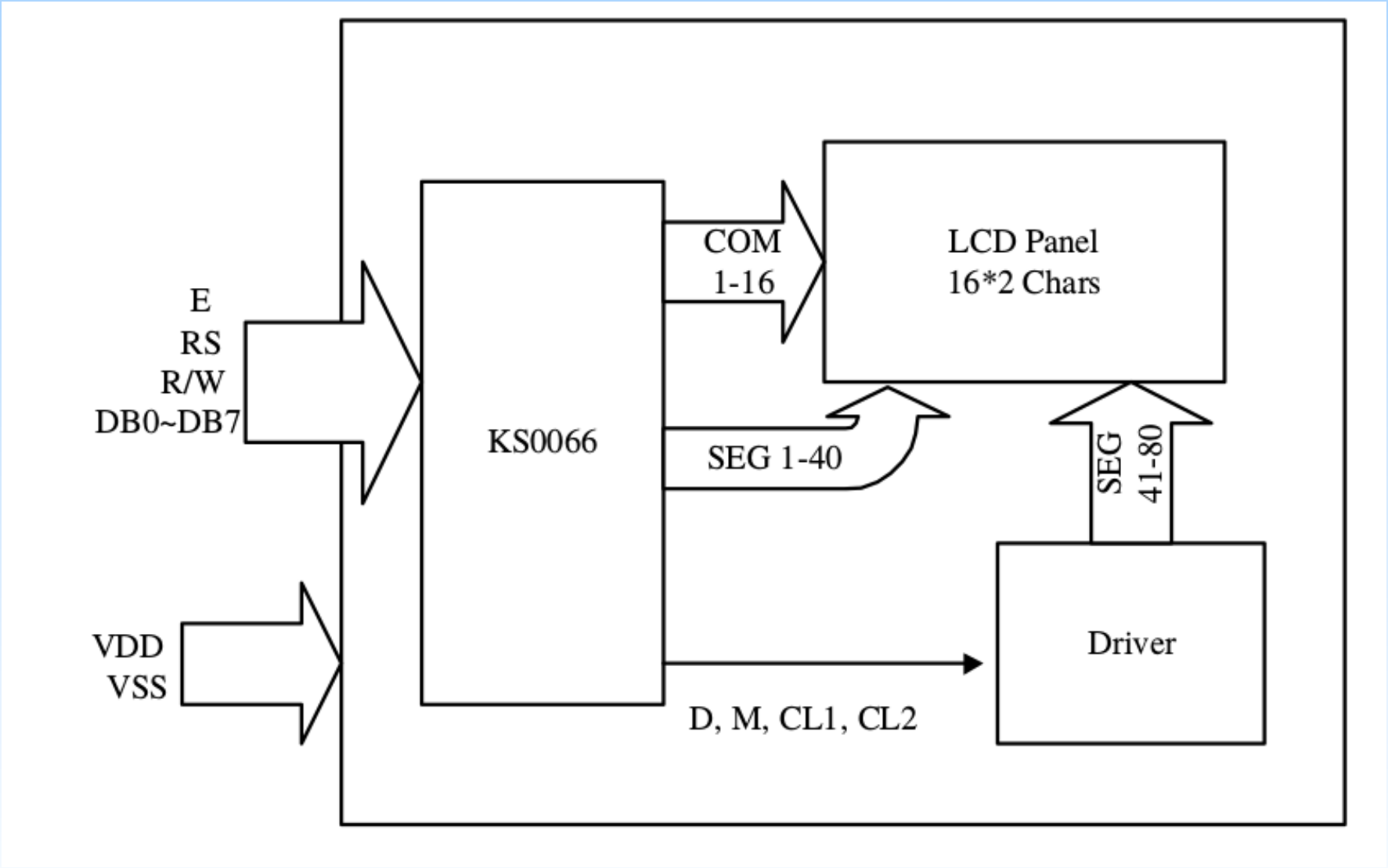
EA871

LCD

Interface de Programação

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

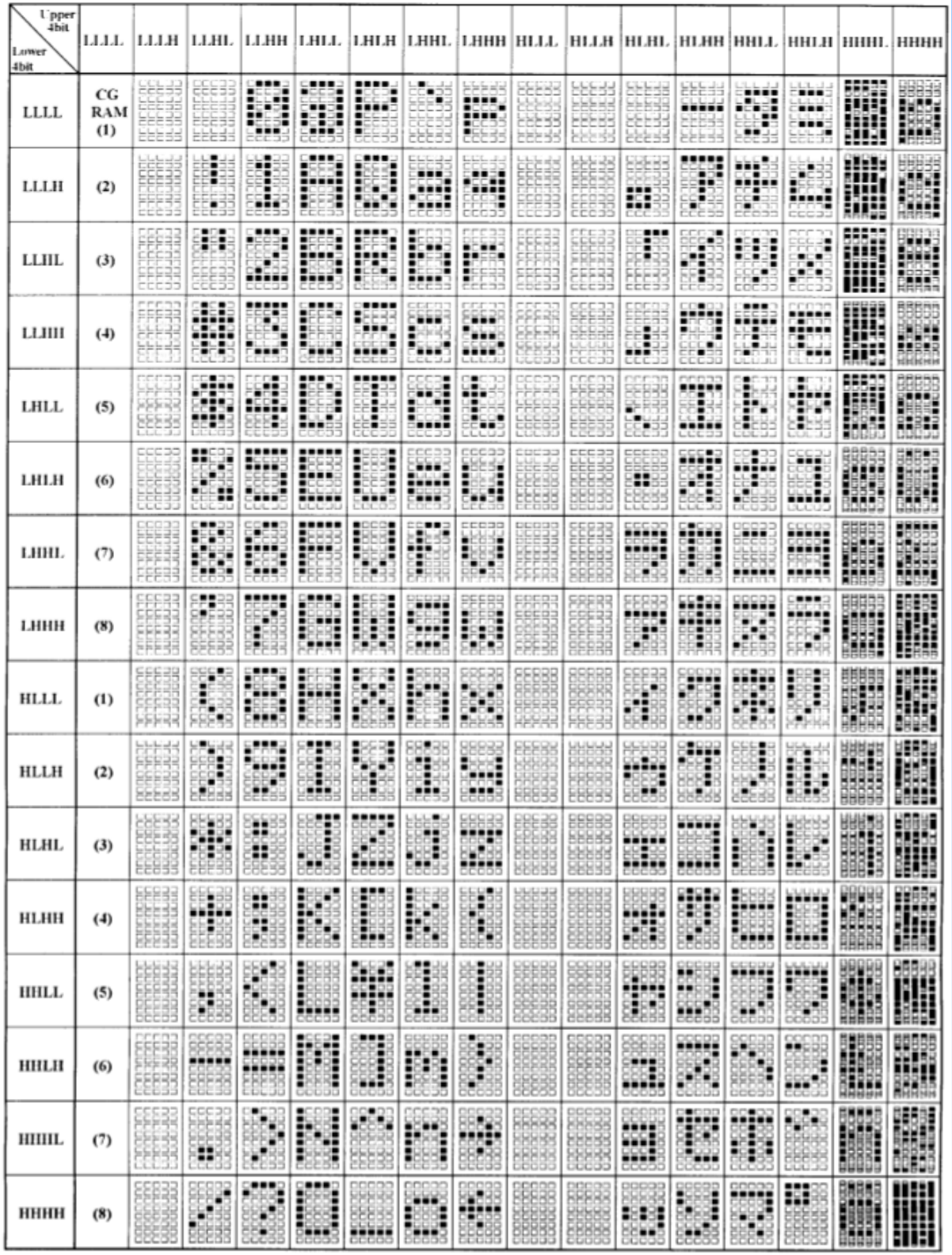
LCD – Diagrama de Blocos



Fonte: ftp://ftp.dca.fee.unicamp.br/pub/docs/ea079/datasheet/AC162A.pdf

Tabela de *bitmaps*:

endereços dos *bitmaps* dos caracteres alfa-numéricos coincidem com os seus códigos ASCII.



LCD – Componentes

- Duas memórias de escrita:
  - **DDRAM:** memória de endereços dos *bitmaps* em exibição no *LCD Panel*.
  - **CGRAM**: memória dos *bitmaps* customizáveis.
- *LCD Panel*: um visor de 2 linhas de 16 matrizes 5x8 para exibição dos *bitmaps* (*LCD Panel*)

DIGIT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1 LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
2 LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

DD RAM Address

- Um controlador para controlar acessos de escrita e de leitura dos dados nas memórias. O endereço de acesso de escrita (conteúdo do **AC**, *Address Counter*) é in/decrementado automaticamente em cada acesso.

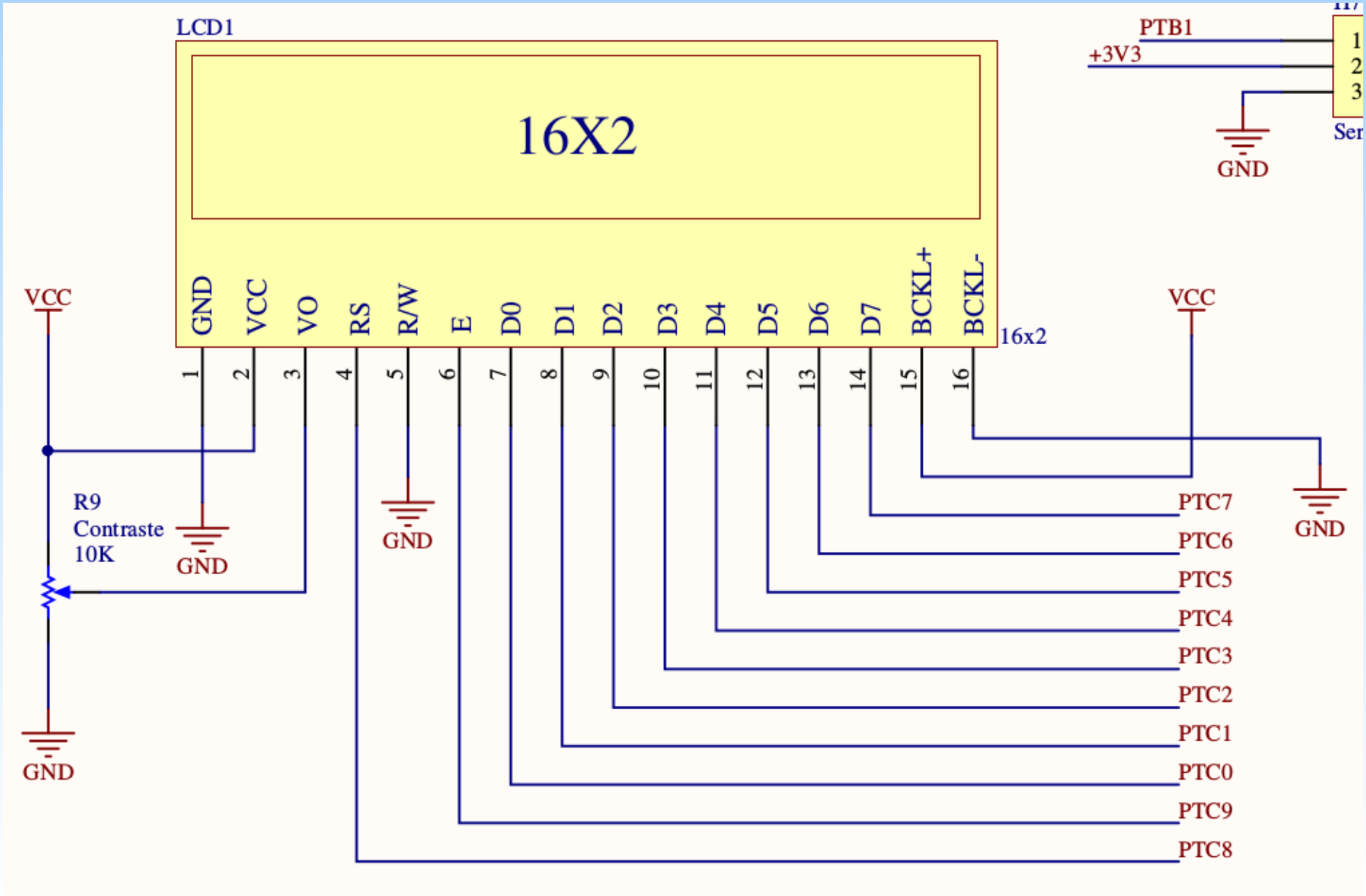
LCD - Interface

No.	Symbol	Function
1	VSS	Ground (0V)
2	VDD	Supply Voltage for Logic (+5V or +3.3V)
3	VO	Contrast Adjustment
4	RS	Data/Instruction Select
5	R/W	Read/Write Select
6	E	Enable Signal
7	DB0	Data Bus
8	DB1	Data Bus
9	DB2	Data Bus
10	DB3	Data Bus
11	DB4	Data Bus
12	DB5	Data Bus
13	DB6	Data Bus
14	DB7	Data Bus

Microcontrolador – LCD

	Microcontrolador	LCD
Funcional	1/0	
Elétrico	3V3/0V; corrente total: 100mA	
Temporal	1 ciclo de instrução ~ 50ns	
Mecânico		

Compatibilidade em Pinagem

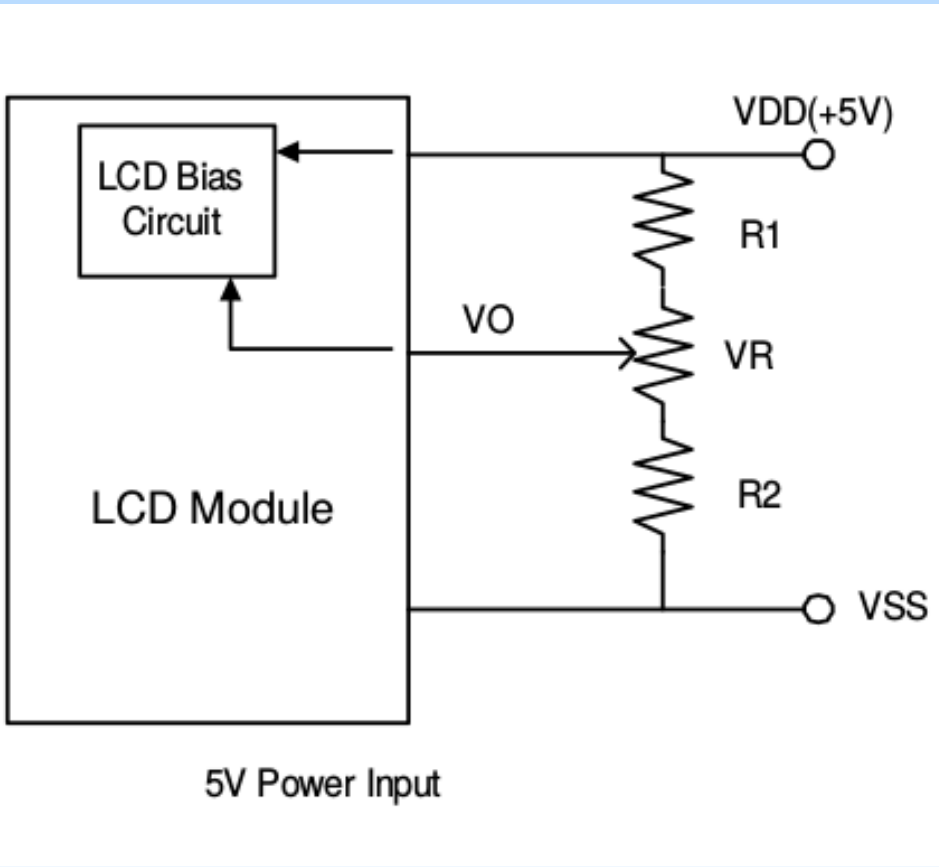


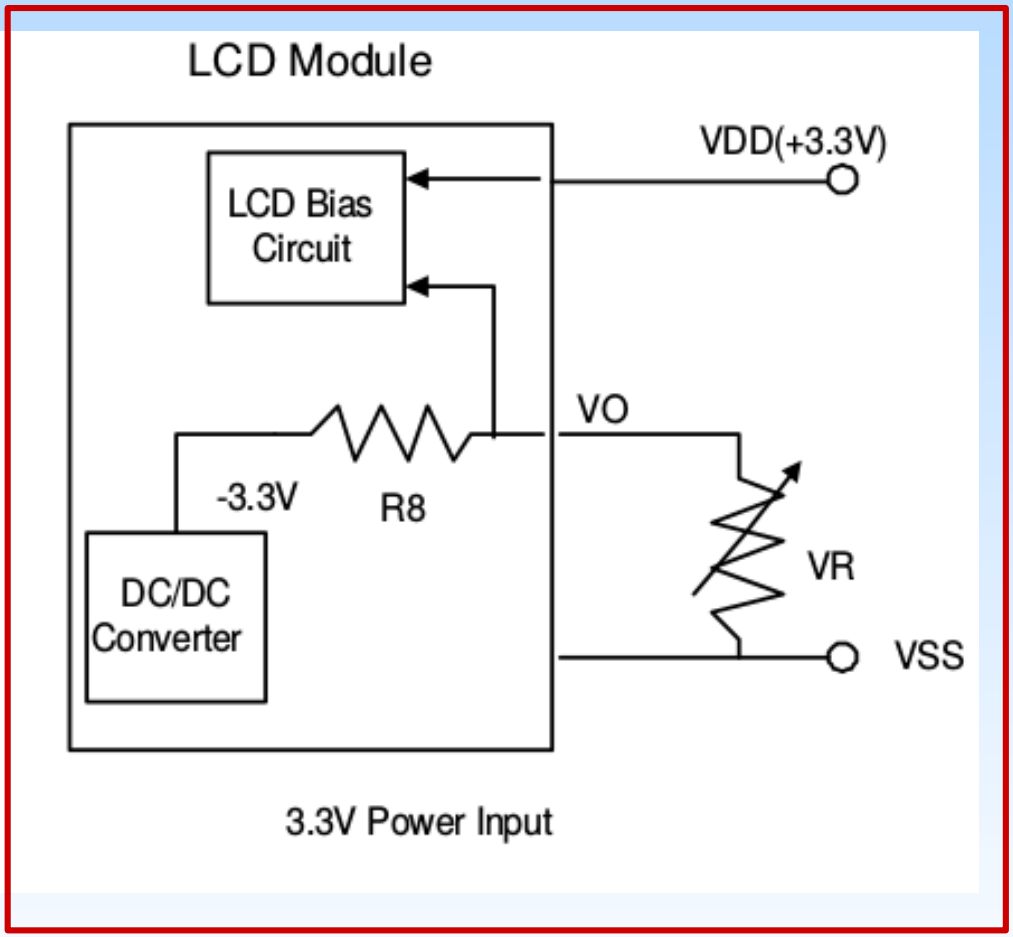
Microcontrolador – LCD

	Microcontrolador	LCD
Funcional	1/0	
Elétrico	3V3/0V; corrente total: 100mA	
Temporal	1 ciclo de instrução ~ 50ns	
Mecânico	10 pinos da porta C (Terra comum)	8 pinos de dados Di e 2 pinos de controle RS e E (Terra comum)

LCD - Alimentação

- Alimentação: 3V3 ou 5V





LCD – Parâmetros Elétricos

Parameter	Symbol	Condition	Min	Typ	Max	Unit	Note
----- Electronic Characteristics -----							
Logic Circuit Supply Voltage	VDD-VSS	--	2.7	--	5.5	V	
LCD Driving Voltage (TN)	VDD-VO	0 °C	4.4	4.7	5.0	V	TN type LCD could only be operated on Normal Temp.
		25 °C	4.2	4.5	4.8		
		50 °C	3.9	4.2	4.5		
LCD Driving Voltage (STN)	VDD-VO	-20 °C	4.75	5.0	5.25	V	0 ~ 50 °C for Normal Temp. type -20 ~ 70 °C for Extended Temp. type
		0 °C	4.75	5.0	5.25		
		25 °C	4.75	5.0	5.25		
		50 °C	4.75	5.0	5.25		
		70 °C	4.75	5.0	5.25		
Input Voltage	VIH	--	0.7 VDD	--	VDD	V	
	VIL	--	VSS	--	0.3 VDD	V	
Logic Supply Current	IDD	VDD = 5V	--	1.0	1.5	mA	

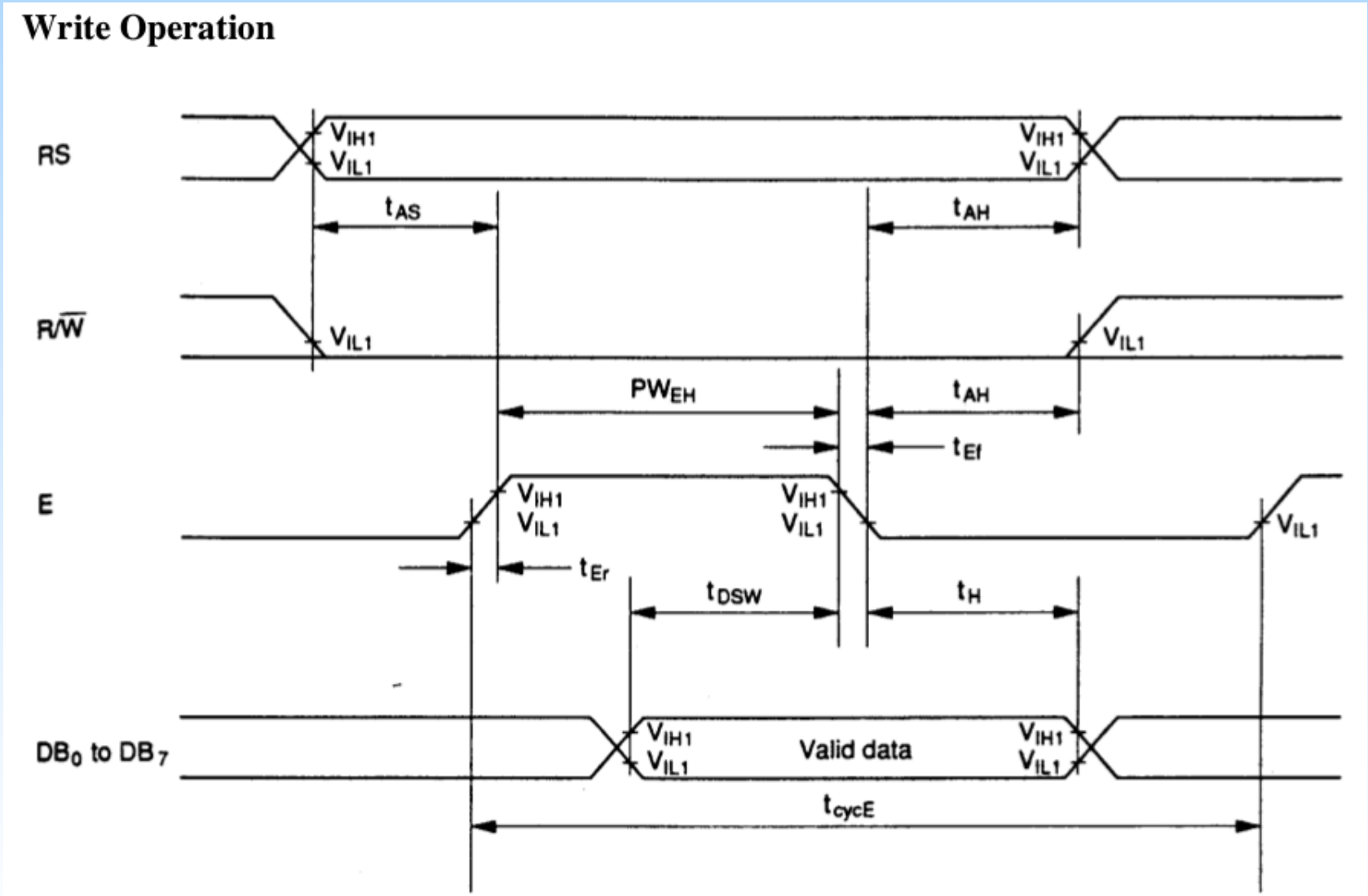
Microcontrolador – LCD

	Microcontrolador	LCD
Funcional	1/0	
Elétrico	3V3/0V; corrente total: 100mA	0.7*VDD/0V; IDD = 1.0mA
Temporal	1 ciclo de instrução ~ 50ns	
Mecânico	10 pinos da porta C (Terra comum)	8 pinos de dados Di e 2 pinos de controle RS e E (Terra comum)

Datasheet de Kinetis: ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0.pdf

LCD – Parâmetros Temporais

- Pino R/\W aterrado → Ciclo de escrita

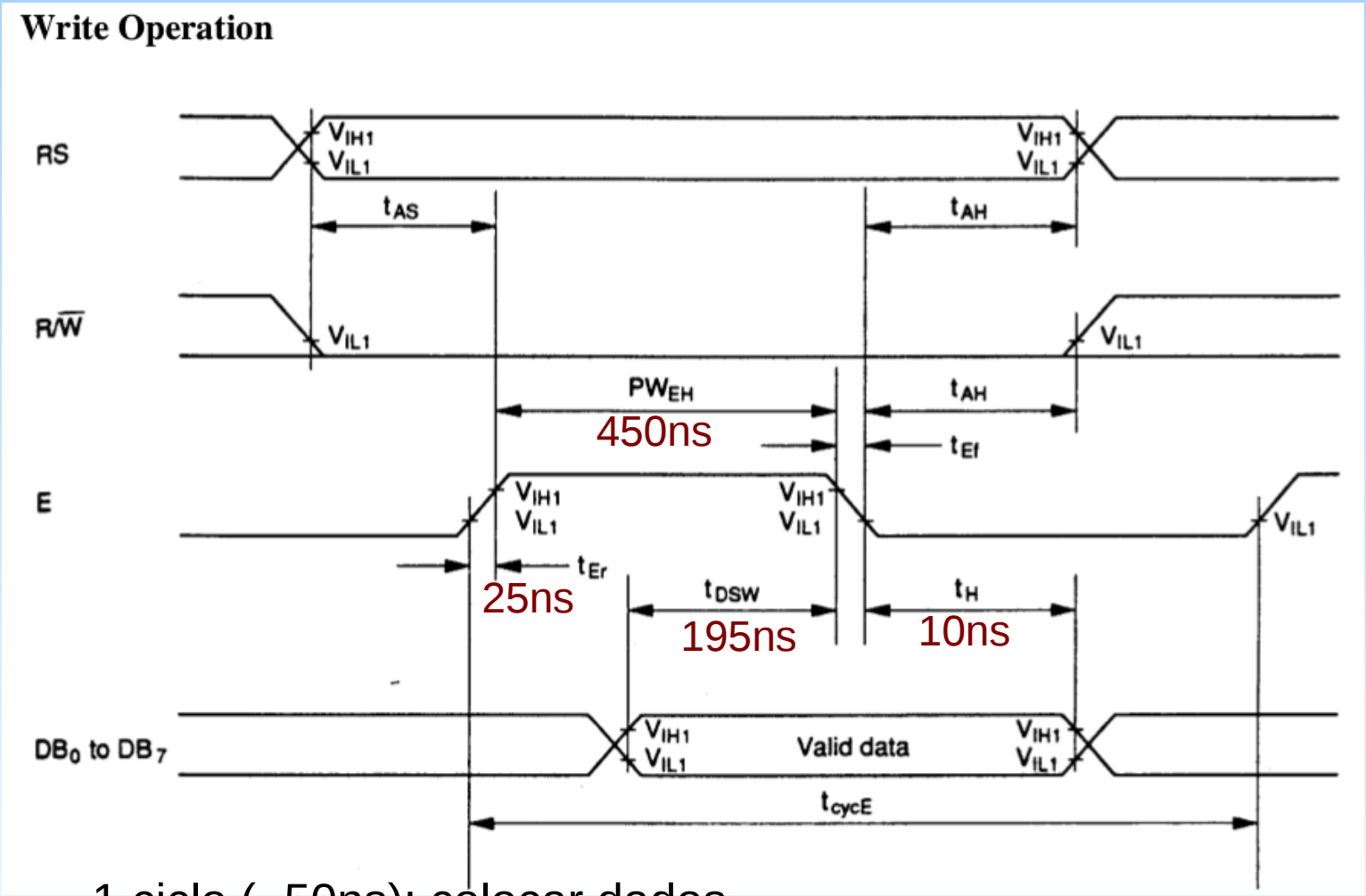


LCD – Parâmetros Temporais

- Pino R/\W aterrado → Ciclo de escrita

Item	Symbol	VDD=5V		VDD=3.3V		Unit
		Min	Max	Min	Max	
Enable cycle time	tcycE	500	--	1000	--	ns
Enable pulse width	PWEH	230	--	450	--	
Enable rise/fall time	tEr,tEf	--	20	--	25	
Address set-up time (RS, R/W to E)	tAS	40	--	60	--	
Address hold time	tAH	10	--	20	--	
Data set-up time	tDSW	80	--	195	--	
Data hold time	tH	10	--	10	--	

LCD – Parâmetros Temporais



1 ciclo (~50ns): colocar dados
1 ciclo (~50ns): colocar E em 1
delay de 450ns
1 ciclo (~50ns): colocar E em 0
1 ciclo (~50ns): remover dados nos pinos (se necessário)

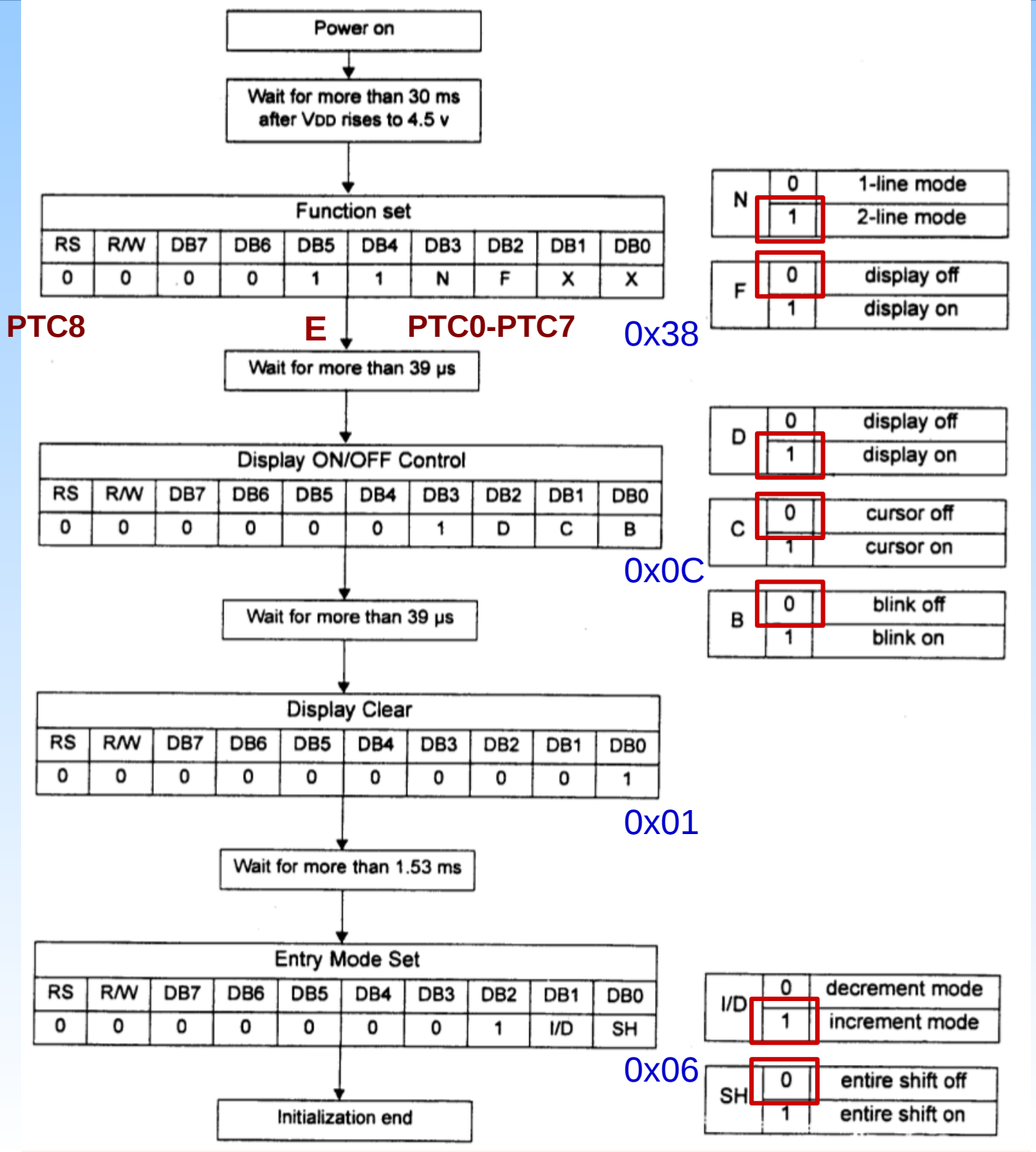
Microcontrolador – LCD

	Microcontrolador	LCD
Funcional	1/0	
Elétrico	3V3/0V; corrente total: 100mA	0.7*VDD/0V; IDD = 1.0mA
Temporal	1 ciclo de instrução ~ 50ns	Inserção de espera para satisfazer as restrições temporais dos sinais
Mecânico	10 pinos da porta C (Terra comum)	8 pinos de dados Di e 2 pinos de controle RS e E (Terra comum)

Microcontrolador – LCD

	Microcontrolador	LCD
Funcional	1/0	Funções programáveis
Elétrico	3V3/0V; corrente total: 100mA	0.7*VDD/0V; IDD = 1.0mA
Temporal	1 ciclo de instrução ~ 50ns	Inserção de espera para satisfazer as restrições temporais dos sinais
Mecânico	10 pinos da porta C (Terra comum)	8 pinos de dados Di e 2 pinos de controle RS e E (Terra comum)

Instruction	Code										Description	E.T.(fosc =270 KHZ)
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write"20H" to DDRAM and set DDRAM address to "00H" from AC	1.53 ms
Return Home	0	0	0	0	0	0	0	0	1	--	Sets DD RAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.53 ms
Entry Mode SET	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.	39 µS
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and blink of cursor (B) on/off control bit.	39 µS
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	--	--	Set cursor moving and display shift control bit, and the direction, without changing of DDRAM data.	39 µS
Function Set	0	0	0	0	1	DL	N	F	--	--	Sets interface data length (DL:8-bit/4-bit), number of display lines (N:2-line/1-line) and , display font type (F:5x11dots/5x8 dost).	39 µS
Set CG RAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Sets CG RAM address in address counter.	39 µS
Set DD RAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Sets DD RAM address in address counter.	39 µS
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0 µS
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Writes data into internal RAM (DD RAM /CG RAM).	43 µS
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Reads data from internal RAM (DD RAM /CG RAM).	43 µS


Power on
Wait for more than 30 ms after VDD rises to 4.5 v
Function set
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 N F X X
PTC8
E
PTC0-PTC7
0x38
N 0 1-line mode 1 2-line mode
F 0 display off 1 display on
Wait for more than 39 µs
Display ON/OFF Control
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 0 0 1 D C B
0x0C
D 0 display off 1 display on
C 0 cursor off 1 cursor on
B 0 blink off 1 blink on
Wait for more than 39 µs
Display Clear
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 0 0 0 0 0 1
0x01
Wait for more than 1.53 ms
Entry Mode Set
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 0 0 0 1 I/D SH
0x06
I/D 0 decrement mode 1 increment mode
SH 0 entire shift off 1 entire shift on
Initialization end

Programação

- **Transferência** de “**Comandos**” para LCD
  - RS ← 0;
  - [D0,D7] ← *byte* de comando;
  - E ← 1;
  - Espera por 450ns
  - E ← 0;
  - Espera pela execução do comando
- **Transferência** de “**Dados**” para LCD
  - RS ← 1;
  - [D0,D7] ← *byte* de dado;
  - E ← 1;
  - Espera por 450ns
  - E ← 0;
  - Espera pelo acesso de dado (43us)
- Diferentes **atrasos** em múltiplos de 2us:
  - 450ns = 0.45us → 1
  - 1,53ms = 1530us → 765
  - 39us → 20
  - 43us → 22

Projeto-Exemplo

- Escrever no meio da primeira linha do LCD: “EA 871”
  - Posicionar o cursor no endereço: AC= 0x05
  - Transferir para LCD a sequência de caracteres ‘E’, ‘A’, ‘ ’, ‘8’, ‘7’, ‘1’
- Laço:
  - Exibir ciclicamente num intervalo de 1s os nomes das cores na segunda linha do LCD: VERMELHO, VERDE, AZUL, AMARELO, CIANO e MAGENTA

Técnica de Programação

- Declaração de vetor de caracteres
  - char str1[] = “EA 871”; //alocados 7 bytes
- Declaração de vetor de vetores de 16 caracteres para sobreescrever o conteúdo de um endereço do DDRAM, evitando “lixo”

```
char lista_cores [6][17] = {"    VERMELHO    ",   //8 caracteres
                            “      VERDE    ",    //5 caracteres
                             "      AZUL      ",   //4 caracteres
                             "     AMARELO    ",    //7 caracteres
                             "      CIANO     ",        //5 caracteres
                             "     MAGENTA    "};   //7 caracteres
```

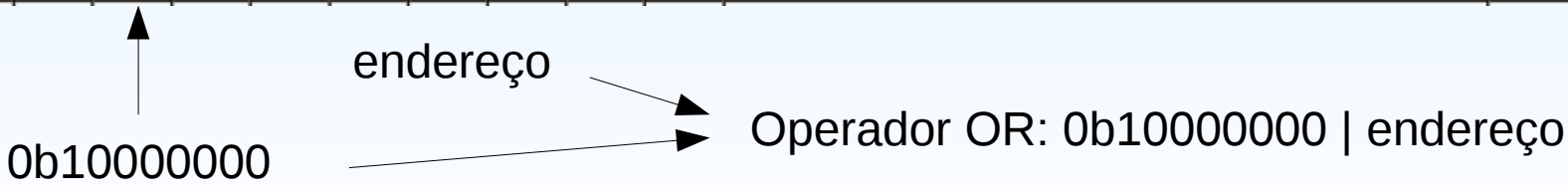
Técnicas de Programação

- Declaração de tipo de dado mais intuitivo

```
typedef enum lcd_RS_tag {
    COMANDO,
    DADO
} lcd_RS_type;
```

- Concatenação de *bits* para codificação dos endereços em DDRAM (memória do *display*)

Set DD RAM Address	0	0	1	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Sets DD RAM address in address counter.	39 µS



Técnicas de Programação

- Modularização dos códigos
  - Inicialização do controle dos pinos D0-D7, RS e E do LCD: **GPIO_initLCD()**
  - Atribuição de valores aos pinos DO-D7 do LCD :
    GPIO_setByte (char c)
  - Atribuição dos valores ao pino RS do LCD:
    GPIO_setRS (lcd_RS_type flag)
  - Habilitação do E: gerar por *software* um pulso de habilitação de ~450ns
    GPIO_enableE ();
  - Inicializa o processamento do LCD
    initLCD();
  - Escreve um *byte* no LCD
    escreveLCD (char c, uint8_t t)
  - Escreve uma mensagem no LCD
    escreveMensagem (uint8_t end, char *mensagem)

Pseudo-Código

```
GPIO_initLCD()
InitLCD()
EscreveMensagem (0x05, “EA 871”);
Laço:
    escreveMensagem(0x40, lista_cores[i]);
    delay (1s);
```

CodeWarrior IDE Development Suite

Informações Adicionais

- Esquemático do shield FEEC871

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/complementos/Esquematico_EA871-Rev3.pdf
- Specifications for LCD Module

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea079/datasheet/AC162A.pdf
- LCD Interfacing Tutorial

  https://www.8051projects.net/lcd-interfacing/index.php
- *Kinetis KL25 Sub-Family datasheet*

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0.pdf
- *KL25 Sub-Family Reference Manual*

  ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
  - Mapa de memória: Seção 4.2 (página 105)
  - SIM: Capítulo 12 (página 192)
  - PORT: Capítulo 11 (página 175)
  - GPIO: Capítulo 41 (página 771)

EA871

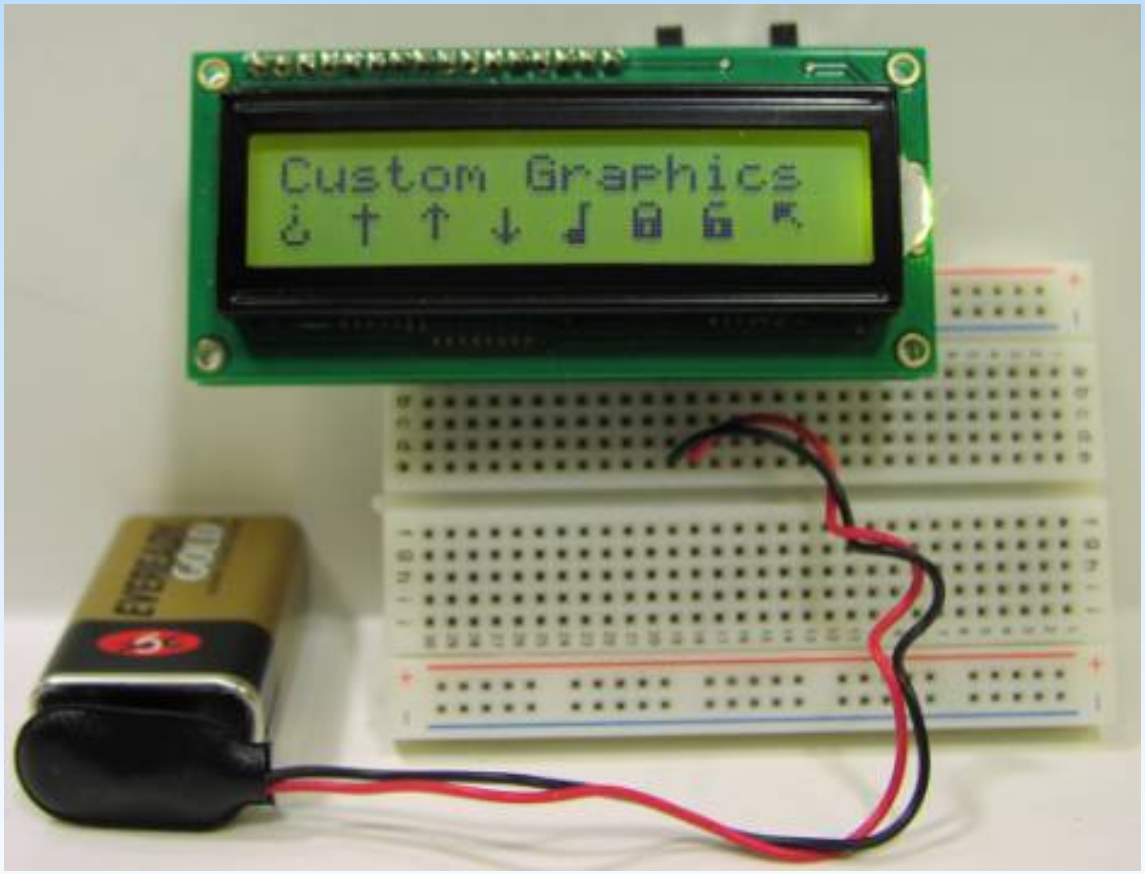
LCD

Bitmaps e Interrupções

Wu Shin – Ting
DCA – FEEC - Unicamp
Segundo Semestre de 2020

Bitmaps

- Figuras formadas por uma matriz de pixels, usualmente quadrados, cujo estado pode ser controlado individualmente.

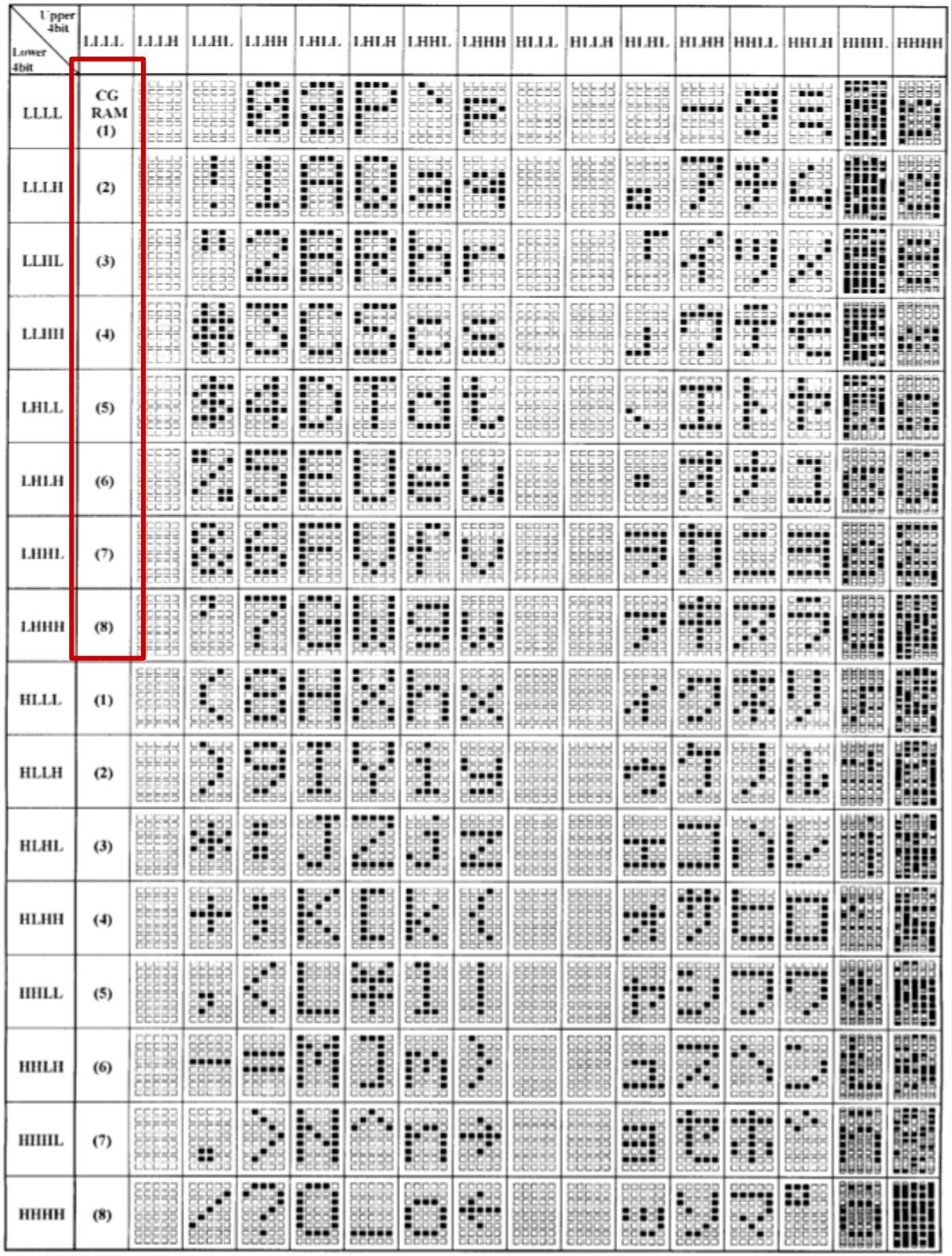


Fonte: https://www.imagesco.com/articles/picdemo/05.html

Conteúdo do DDRAM (memória do *display* do LCD): endereços dos *bitmaps* na tabela.

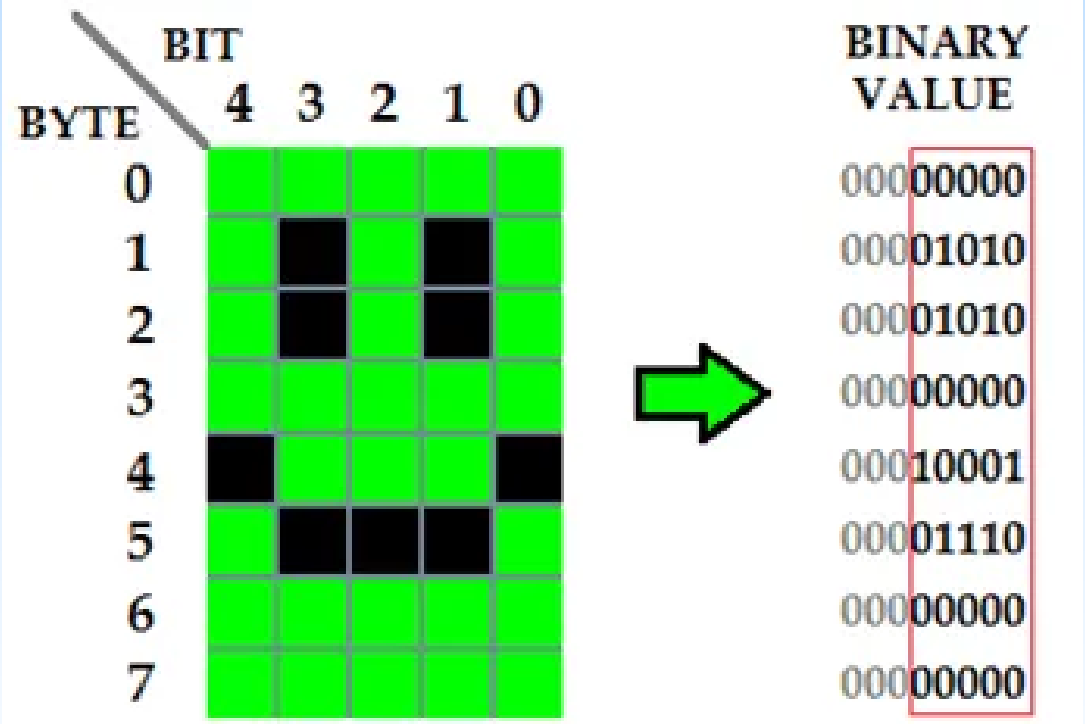
Endereços dos primeiros 127 *bitmaps* correpondem aos respectivos códigos ASCII.

Bitmaps de endereços 0b00000000 a 0b00000111 são customizáveis.



Bitmaps em CGRAM

- Cada *bitmap* ocupa 8 *bytes*.
- Cada endereço do *byte* pode ser determinado a partir do endereço do *bitmap* <end>, de 0 a 7, isto é

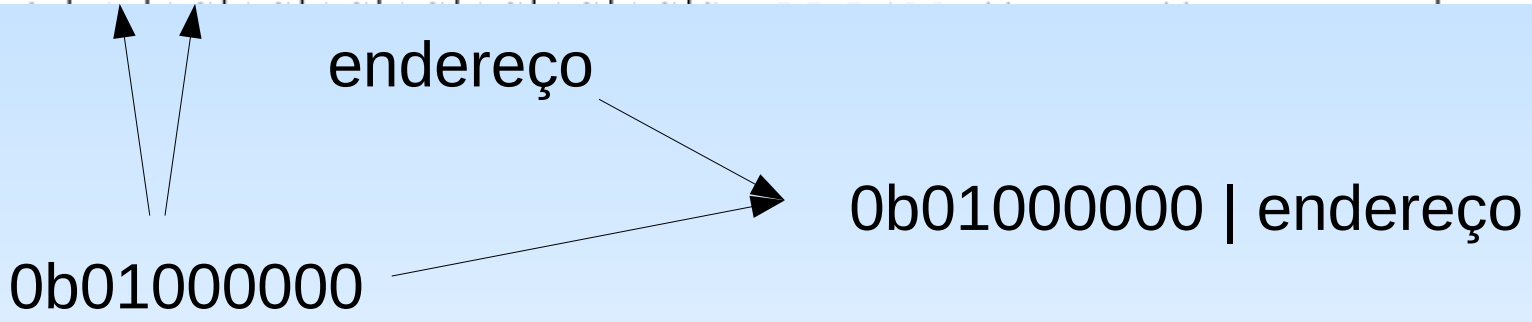


end*8
end*8+1
end*8+2
end*8+3
end*8+4
end*8+5
end*8+6
end*8+7

CGRAM: Especificação do endereço

- *Bits* 7 e 6: 0b01

Set CG RAM Address	0	0	0	1	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0	Sets CG RAM address in address counter.	39 µS



Pseudocódigo de Definição de um *Bitmap* em <end>

```
GPIO_setRS (COMANDO)
escreveLCD(b01000000 | (end*8), 39us);]
GPIO_setRS(DADO)
for (i=0; i<8; i++)
    escreveLCD(bitmap[i], 43us);
```

Técnica de Programação

- **Interrupção** = atendimentos emergenciais
- Processamento de LCD (instruções na ordem de 40us) x processamento por instrução do processador (instruções na ordem de 50ns)
  - Exibição de 16 caracteres ~ 16x43us ~ 640us!
  - **Evitar que o processador fique "preso" no atendimento de um LCD dentro de uma rotina de interrupção.**
    - **Interrupções aninhadas** (*nested*): setar o atendimento do LCD em menor prioridade.
    - Programar o sistema para que a exibição das mensagens ocorram no "**estado de espera**" do processador.

Projeto-exemplo

- Construir 3 diferentes *bitmaps* e mostrar cada um, junto com o seu nome na segunda linha do LCD, ao acionarmos as botoeiras NMI, IRQ5 e IRQ12.
- Uma solução:
  - processamento dos eventos das botoeiras por interrupção.
  - Processar o LCD no **estado de espera**

Pseudocódigo

- GPIO_initLedRGB(); inicializa os registradores de controle dos *leds*
- GPIO_initSwitches(): inicializa os registradores de controle das chaves
- GPIO_enableSwInterrupt(): habilita para interrupções
- **Laço de espera:**
  - **Se bitmap contém valor válido, então processar LCD**

- Rotina de Serviço (**PORTA_IRQHandler** declarada em **kinitis_sysinit.c**):
  - Se (PORTA_PCR4_ISF == 1) então GPIO_LedRGB(cor[VERMELHO]); **bitmap**=bitmap1; Limpa PORTA_PCR4_ISF; retorna;
  - Se (PORTA_PCR5_ISF == 1) então GPIO_LedRGB(cor[VERDE]); **bitmap**=bitmap2;

    Limpa PORTA_PCR5_ISF; retorna;
  - Se (PORTA_PCR12_ISF == 1) então GPIO_LedRGB(cor[AZUL]); **bitmap**=bitmap3;

    Limpa PORTA_PCR12_ISF; retorna;

Técnica de Programação

- Como “instruir” o compilador que as variáveis bitmap da rotina de serviço e da função main tenham o mesmo endereço da memória?
  - declaração de que a variável seja **global**:
    - dentro de um mesmo arquivo:
      - Declarar fora do escopo das funções
    - entre diferentes arquivos
      - Declará-la num dos arquivos e referenciá-la com uso do qualificador **extern** em outros arquivos

Técnica de Programação

- Declaração de um novo tipo de dados para facilitar a interpretação dos tipos de *bitmaps*

```
typedef enum bitmap_tag {
    HAPPY,
    HEART,
    BELL,
    NONE
} bitmap_type;
```

CodeWarrior IDE Development Suite

Informações Adicionais

- LCD Interfacing Tutorial: CGRAM Creating custom character

  https://www.8051projects.net/lcd-interfacing/lcd-custom-character.php