



Universidade Tecnológica Federal do Paraná – UTFPR
Bacharelado em Ciência da Computação

BCC34C – Sistemas Microcontrolados

Frank Helbert Borsato

Acionando LCD

- **Os Displays de Cristal líquido**
- **LCD, que significam “LiquidCrystal Display”**
 - São interfaces que utilizam uma tecnologia moderna para representar letras, números e símbolos advindos de microprocessadores ou de microcontroladores
 - Estes módulos podem ser gráficos ou a caractere (alfanuméricos)
- **Os LCDs comuns, tipo caractere, são especificados em número de colunas x linhas**
 - 16 × 2
 - » Utilizaremos o modelo 16 x 2 (16 colunas e 2 linhas)
 - 16 × 1
 - 20 × 2
 - 20 × 4
 - 8 × 2



Acionando LCD

- Para 'desenhar' as letras, números e outros símbolos
- Cada dígito é composto por uma matriz, fabricado tipicamente com 5 colunas e 8 linhas
 - A última é utilizada para o cursor, por este motivo alguns usuários o atribuem o tamanho de 5x7 ao invés de 5x8

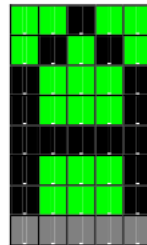


Figura 1) matriz 5x8 dos caracteres

Acionando LCD

- **A arquitetura interna do LCD – CI controlador HD44780**

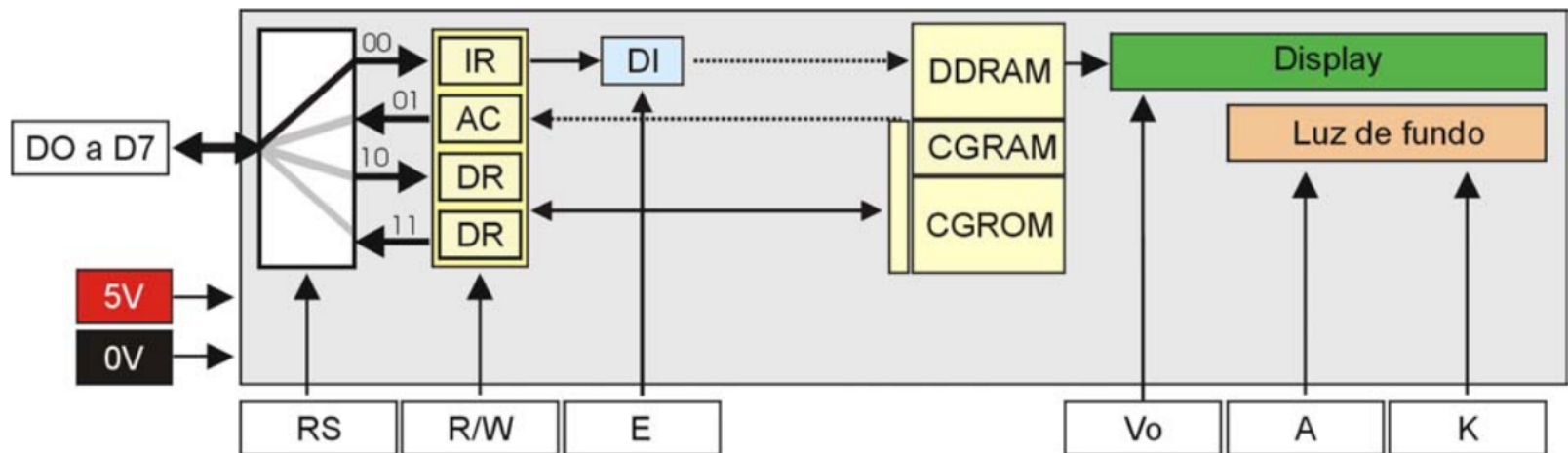


Figura 3) arquitetura interna hipotética (proposta pelo autor)

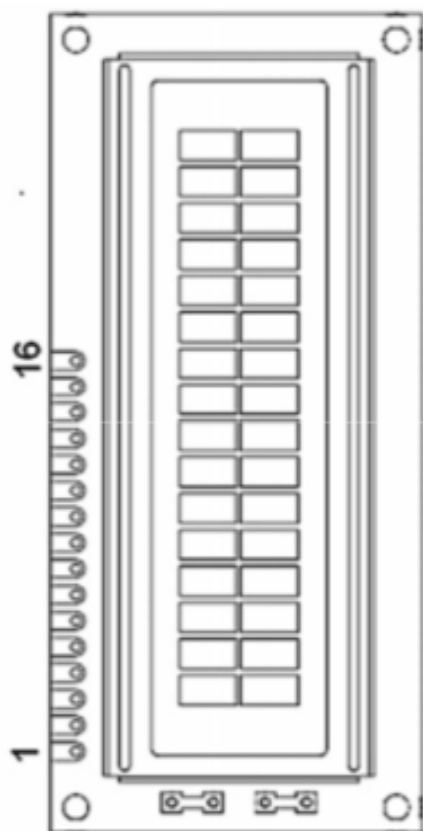


Diagrama LCD do livro texto



LCD real

Tab. B1: Pinagem de um LCD 16 × 2.



Pino	Função	Descrição
1	Alimentação	VSS (GND)
2	Alimentação	VCC
3	VEE	Tensão para ajuste do contraste do LCD
4	RS	<i>Register Select</i> : 1 = dado, 0 = instrução
5	R/W	<i>Read/Write</i> : 1 = leitura, 0 = escrita
6	E	<i>Enable</i> : 1 = habilita, 0 = desabilita
7	DB0	Barramento de dados
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+ (A)	Anodo do LED de iluminação de fundo
16	LED - (K)	Catodo do LED de iluminação de fundo

- **A arquitetura interna do LCD – CI controlador HD44780**



E: Enable (1 = habilita, 0 = desabilita)

Acionando LCD

- A arquitetura interna do LCD – CI controlador HD44780

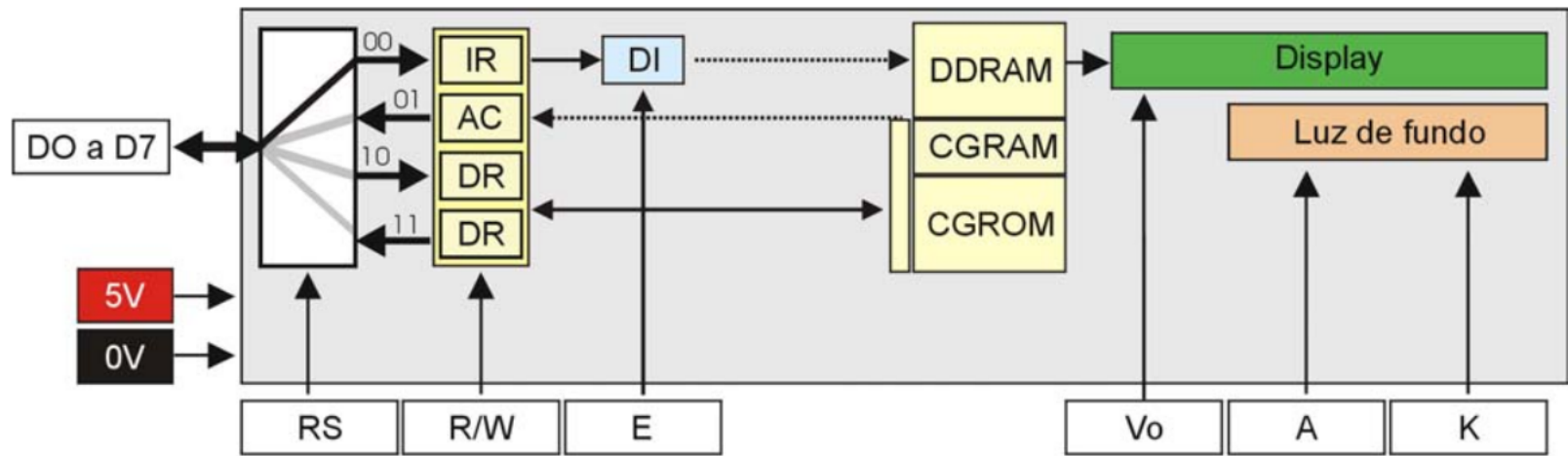


Figura 3) arquitetura interna hipotética (proposta pelo autor)

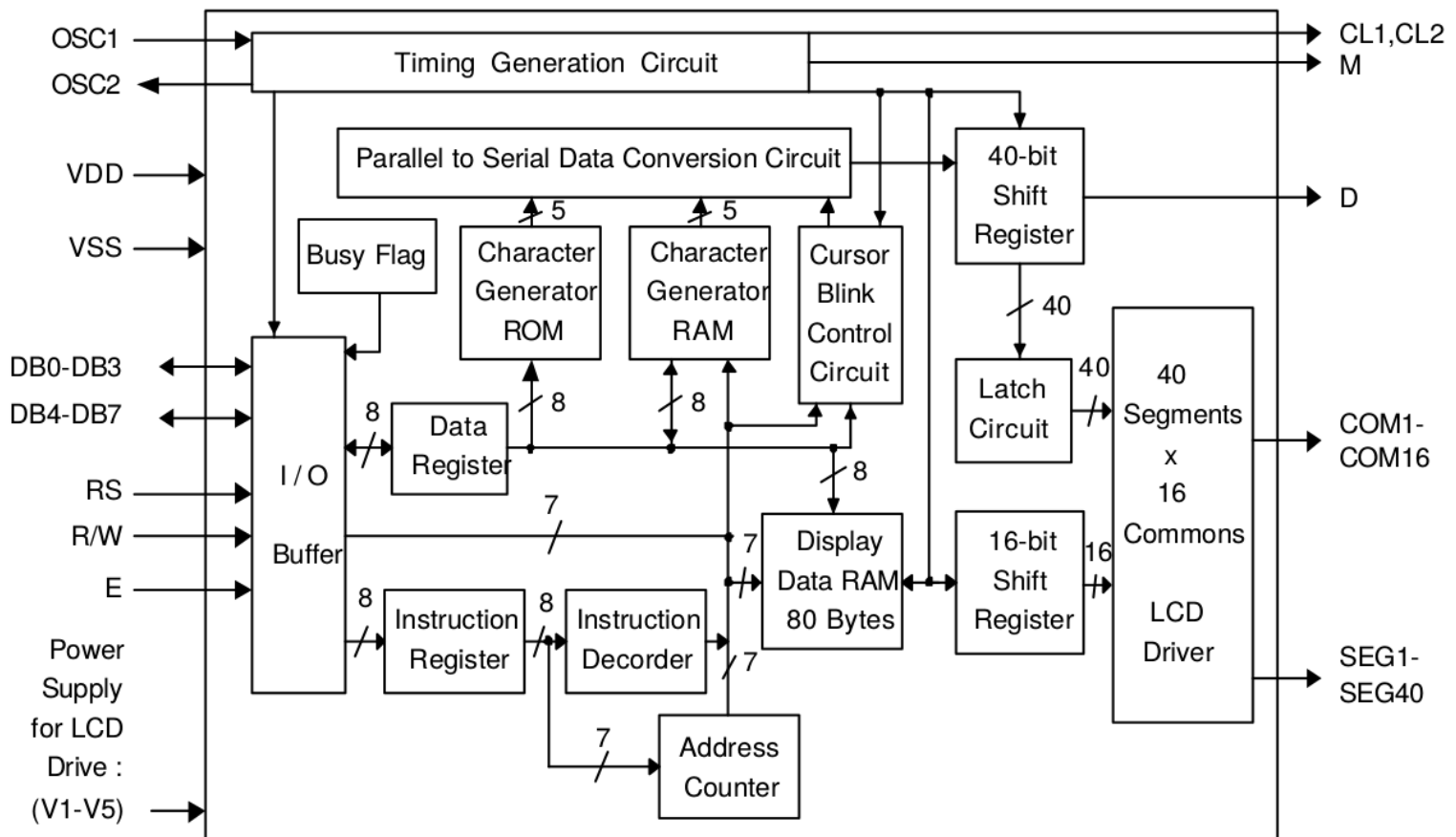
O registrador de instruções (IR - Instruction Register)

O registrador contador de endereço é chamado de AC (Address Counter)

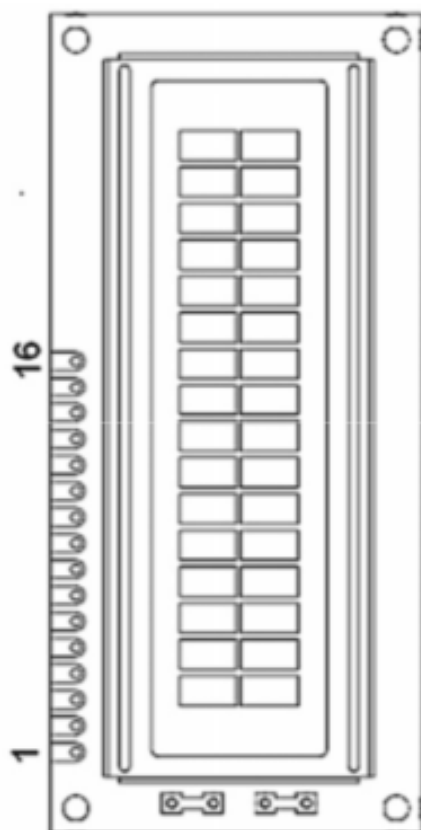
O registrador de dados, DR (Data Register)

Acionando LCD

- A arquitetura interna do LCD – CI controlador HD44780



Tab. B1: Pinagem de um LCD 16 × 2.



Pino	Função	Descrição
1	Alimentação	VSS (GND)
2	Alimentação	VCC
3	VEE	Tensão para ajuste do contraste do LCD
4	RS	<i>Register Select</i> : 1 = dado, 0 = instrução
5	R/W	<i>Read/Write</i> : 1 = leitura, 0 = escrita
6	E	<i>Enable</i> : 1 = habilita, 0 = desabilita
7	DB0	Barramento de dados
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+ (A)	Anodo do LED de iluminação de fundo
16	LED - (K)	Catodo do LED de iluminação de fundo

Acionando LCD

- Os controladores de display de cristal líquido possuem um bloco de memória que totaliza 384 bytes efetivos
- Essa memória se divide em três áreas:
 - Uma somente leitura (CGROM) – não volátil
 - Duas de escrita/leitura (DDRAM e CGRAM) - volátil

Acionando LCD

- **A CGROM (Character Generator Read Only Memory) não pode ser modificada pelo programador**
- **Possui 192 caracteres pré-programados**
- **Endereçados de 20h a 7Fh e de A0h a FFh**
 - Um código ASCII é enviado para o LCD, através do registrador DR
 - » **O controlador utiliza este código como endereço e verifica na memória CGROM o mapa de bits correspondente aquele caractere**
 - » **O caractere é enviado ao display através da DDRAM**
 - » Por este motivo a CGROM é chamada de geradora de caracteres
 - Por exemplo, endereço 41h (65)₁₀ corresponde à letra “A”

NIBBLE BAIXO \ NIBBLE ALTO		0_	1_	2_	3_	4_	5_	6_	7_	8_	9_	A_	B_	C_	D_	E_	F_
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
_0	xxxx 0000	CG RAM (1)				00PzP											
_1	xxxx 0001	CG RAM (2)				10Qa9											
_2	xxxx 0010	CG RAM (3)				"2BRbr											
_3	xxxx 0011	CG RAM (4)				#3CScs											
_4	xxxx 0100	CG RAM (5)				\$4DTdt											
_5	xxxx 0101	CG RAM (6)				%5EUeu											
_6	xxxx 0110	CG RAM (7)				&6FVfv											
_7	xxxx 0111	CG RAM (8)				'7Gwsw											
_8	xxxx 1000	(1)				(8HXhx											
_9	xxxx 1001	(2))9IYiy											
_A	xxxx 1010	(3)				*:JZjz											
_B	xxxx 1011	(4)				+;Klk{											
_C	xxxx 1100	(5)				,<L*ll											
_D	xxxx 1101	(6)				-=MIm}											
_E	xxxx 1110	(7)				.>N^n+											
_F	xxxx 1111	(8)				/?O_lo+											

Acionando LCD

- **DDRAM**

- **O controlador possui uma memória RAM de dados de 80 bytes**
- **Endereçados de (00h a 67h)**
 - Este é o mapa de memória mais comum para os 80 bytes de DDRAM no controlador HD44780

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Acionando LCD

- **DDRAM**

- Como você pode ver, a DDRAM consiste em duas linhas de memória com um intervalo um tanto misterioso no endereçamento quando vai da primeira linha de memória para o segundo
 - A primeira linha tem 40 locais de armazenamento identificados pelos endereços 00h a 27h
 - A segunda linha tem outros 40 locais de armazenamento identificados pelos endereços 40h até 67h.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Acionando LCD

- **DDRAM**

- Como você pode ver, a DDRAM consiste em duas linhas de memória com um intervalo um tanto misterioso no endereçamento quando vai da primeira linha de memória para o segundo
 - A lacuna misteriosa é devido a considerações resultantes da multiplexação da exibição
 - O endereçamento DDRAM usa endereçamento de sete bits e o bit mais alto significa que linha de memória está envolvida
 - Se você compara os endereços na primeira linha com aqueles logo abaixo na segunda linha você verá que a única diferença é um bit.

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

Acionando LCD

• DDRAM

- O bit mais significativo do endereçamento da RAM sempre é 1
- Endereçados de (00h a 67h) para (80-E7)

5.2.8. Set display data RAM address

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	1	a	a	a	a	a	a	a

It sets Display Data RAM Address (aaaaaaa)₂ to the Address Counter.

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0
8				0			

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0
C				0			

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0
0				0			

7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0
4				0			

Acionando LCD

- **DDRAM**

- O bit mais significativo do endereçamento da RAM sempre é 1
- Endereços de (00h a 67h) para (80-E7)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Modo (Rs=0 e RW=0)									IR
Limpa o display (DDRAM) e posiciona o cursor no início	0	0	0	0	0	0	0	1	01
Posiciona cursor no início (sem apagar DDRAM)	0	0	0	0	0	0	1	x	02-03
Desloca cursor para esquerda após escrever caractere	0	0	0	0	0	1	0	0	04
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	0	1	05
Desloca cursor para direita após escrever caractere	0	0	0	0	0	1	1	0	06
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	1	1	07
Desliga display e cursor, mantendo os dados na DDRAM	0	0	0	0	1	0	0	0	08
Desliga display, dados permanecem na DDRAM, cursor pisca	0	0	0	0	1	0	0	1	09
Desliga display e liga cursor fixo	0	0	0	0	1	0	1	0	0A
Desliga display, mantendo os dados na DDRAM, liga cursor	0	0	0	0	1	0	1	1	0B
Liga o display e esconde o cursor piscante	0	0	0	0	1	1	0	0	0C
Liga o display e o cursor fica piscando	0	0	0	0	1	1	0	1	0D
Liga o display e o cursor fica fixo	0	0	0	0	1	1	1	0	0E
Liga o display e o cursor fica alternante	0	0	0	0	1	1	1	1	0F
Desloca cursor para esquerda e decrementa AC	0	0	0	1	0	0	x	x	10-13
Desloca cursor para direita e incrementa AC	0	0	0	1	0	1	x	x	14-17
Desloca mensagem para esquerda e cursor acompanha	0	0	0	1	1	0	x	x	18-1B
Desloca mensagem para direita e cursor acompanha	0	0	0	1	1	1	x	x	1C-1F
Interface 4 bits, display de 1 linha e matriz 5x8	0	0	1	0	0	0	x	x	20-23
Interface 4 bits, display de 1 linha e matriz 5x11	0	0	1	0	0	1	x	x	24-27
Interface 4 bits, display de 2 linha e matriz 5x8	0	0	1	0	1	0	x	x	28-2B
Interface 4 bits, display de 2 linha e matriz 5x11	0	0	1	0	1	1	x	x	2C-2F
Interface 8 bits, display de 1 linha e matriz 5x8	0	0	1	1	0	0	x	x	30-33
Interface 8 bits, display de 1 linha e matriz 5x11	0	0	1	1	0	1	x	x	34-37
Interface 8 bits, display de 2 linha e matriz 5x8	0	0	1	1	1	0	x	x	38-3B
Interface 8 bits, display de 2 linha e matriz 5x11	0	0	1	1	1	1	x	x	3C-3F
Endereços para escrever na CGRAM	0	1	AC5	AC4	AC3	AC2	AC1	AC0	40-7F
Endereços para escrever/ler na DDRAM	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	80-FF

Tabela 4) código das instruções que comandam o LCD

Acionando LCD

- **DDRAM**

- **Ela é dividida em blocos, que se referem às linhas do display**

- A primeira linha inicia em 80h e vai até A7h

- A segunda linha inicia em C0h e vai até E7h

- » **Compondo um total de 64 caracteres cada linha (máximo)**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Endereços dos seguimentos DDRAM LCD 16 x 2 (Linha 1: 80 a 8F, Linha 2: C0 a CF)

Acionando LCD

- **DDRAM**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
Linha 3	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
Linha 4	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Fig. 5.22 – Endereços para a escrita num LCD 20 × 4.

Acionando LCD

- **CGRAM (Character Generator RAM)**
- **Área na qual o programador pode definir caracteres especiais que não constam na tabela ASCII**
 - Possui tipicamente 8 matrizes para compor os caracteres customizáveis
 - Cada matriz utiliza 8 bytes totalizando uma memória de 64 bytes acessados através de um conjunto de endereços
 - » $2^6 = 64$ (utiliza 6 bits de endereçamento)

5.2.7. Set character generator RAM address

	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Code	0	0	0	1	a	a	a	a	a	a

It sets Character Generator RAM Address $(aaaaaa)_2$ to the Address Counter.

Modo (Rs=0 e RW=0)									IR
Limpa o display (DDRAM) e posiciona o cursor no início	0	0	0	0	0	0	0	1	01
Posiciona cursor no início (sem apagar DDRAM)	0	0	0	0	0	0	1	x	02-03
Desloca cursor para esquerda após escrever caractere	0	0	0	0	0	1	0	0	04
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	0	1	05
Desloca cursor para direita após escrever caractere	0	0	0	0	0	1	1	0	06
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	1	1	07
Desliga display e cursor, mantendo os dados na DDRAM	0	0	0	0	1	0	0	0	08
Desliga display, dados permanecem na DDRAM, cursor pisca	0	0	0	0	1	0	0	1	09
Desliga display e liga cursor fixo	0	0	0	0	1	0	1	0	0A
Desliga display, mantendo os dados na DDRAM, liga cursor	0	0	0	0	1	0	1	1	0B
Liga o display e esconde o cursor piscante	0	0	0	0	1	1	0	0	0C
Liga o display e o cursor fica piscando	0	0	0	0	1	1	0	1	0D
Liga o display e o cursor fica fixo	0	0	0	0	1	1	1	0	0E
Liga o display e o cursor fica alternante	0	0	0	0	1	1	1	1	0F
Desloca cursor para esquerda e decrementa AC	0	0	0	1	0	0	x	x	10-13
Desloca cursor para direita e incrementa AC	0	0	0	1	0	1	x	x	14-17
Desloca mensagem para esquerda e cursor acompanha	0	0	0	1	1	0	x	x	18-1B
Desloca mensagem para direita e cursor acompanha	0	0	0	1	1	1	x	x	1C-1F
Interface 4 bits, display de 1 linha e matriz 5x8	0	0	1	0	0	0	x	x	20-23
Interface 4 bits, display de 1 linha e matriz 5x11	0	0	1	0	0	1	x	x	24-27
Interface 4 bits, display de 2 linha e matriz 5x8	0	0	1	0	1	0	x	x	28-2B
Interface 4 bits, display de 2 linha e matriz 5x11	0	0	1	0	1	1	x	x	2C-2F
Interface 8 bits, display de 1 linha e matriz 5x8	0	0	1	1	0	0	x	x	30-33
Interface 8 bits, display de 1 linha e matriz 5x11	0	0	1	1	0	1	x	x	34-37
Interface 8 bits, display de 2 linha e matriz 5x8	0	0	1	1	1	0	x	x	38-3B
Interface 8 bits, display de 2 linha e matriz 5x11	0	0	1	1	1	1	x	x	3C-3F
Endereços para escrever na CGRAM	0	1	AC5	AC4	AC3	AC2	AC1	AC0	40-7F
Endereços para escrever/ler na DDRAM	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	80-FF

Tabela 4) código das instruções que comandam o LCD

Acionando LCD

- **CGRAM (Caractere Generator RAM)**
- Cada matriz utiliza 8 bytes totalizando uma memória de 64 bytes acessados através de um conjunto de endereços

ASCII	Endereço (Hexadecimal)	CGRAM
0	40 a 47	
1	48 a 4F	
2	50 a 58	
3	58 a 5F	
4	60 a 67	
5	68 a 6F	
6	70 a 77	
7	78 a 7F	

Tabela 3) endereços da CGRAM

Acionando LCD

- **CGRAM (Caractere Generator RAM)**
- Os dados da CGRAM são apresentados em um mapa de bits de 8 bytes
 - Utilizam 7, com 5 bits cada, sendo 1 byte reservado para o cursor

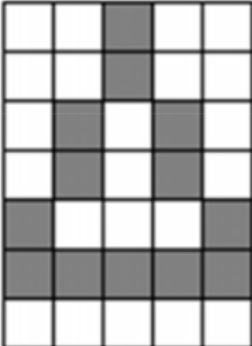
Endereço da CGRAM	Mapa de bits	Dado
0x48		0b00100
0x49		0b00100
0x4A		0b01010
0x4B		0b01010
0x4C		0b10001
0x4E		0b11111
0x4F		0b00000

Fig. 5.24 – Gravação do símbolo Δ na CGRAM, matriz 5×7 . Esse caractere será selecionado pelo código 0x01.

Acionando LCD

- **O registrador de instruções (IR - Instruction Register):**
 - **Comanda o display de acordo com as operações de configuração e instruções**

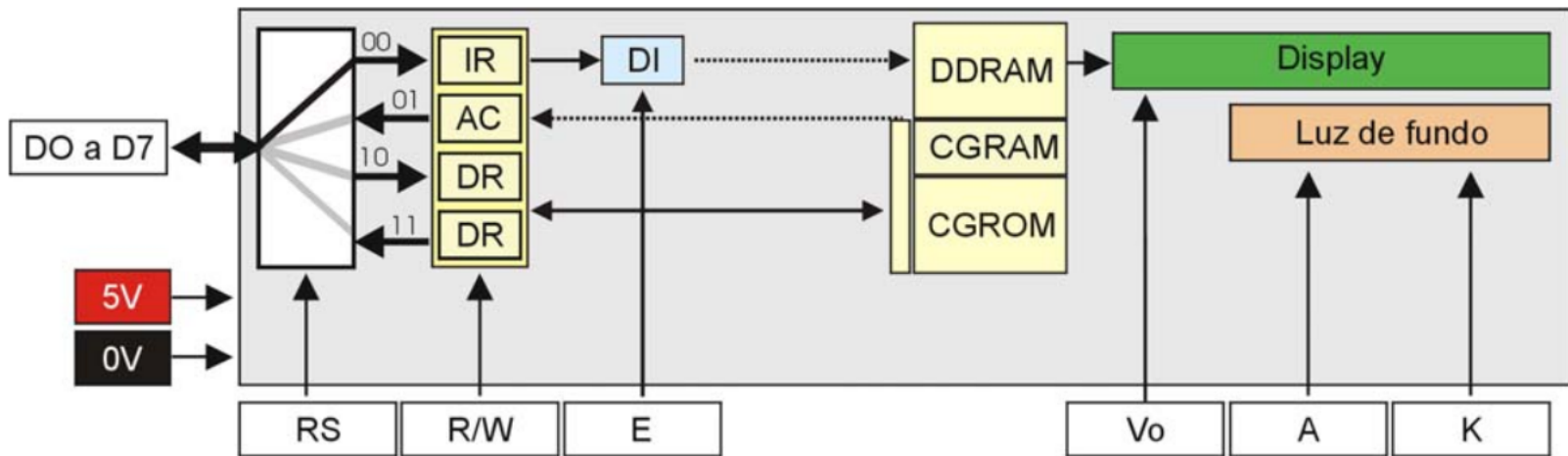


Figura 3) arquitetura interna hipotética (proposta pelo autor)

Modo (Rs=0 e RW=0)									IR
Limpa o display (DDRAM) e posiciona o cursor no início	0	0	0	0	0	0	0	1	01
Posiciona cursor no início (sem apagar DDRAM)	0	0	0	0	0	0	1	x	02-03
Desloca cursor para esquerda após escrever caractere	0	0	0	0	0	1	0	0	04
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	0	1	05
Desloca cursor para direita após escrever caractere	0	0	0	0	0	1	1	0	06
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	1	1	07
Desliga display e cursor, mantendo os dados na DDRAM	0	0	0	0	1	0	0	0	08
Desliga display, dados permanecem na DDRAM, cursor pisca	0	0	0	0	1	0	0	1	09
Desliga display e liga cursor fixo	0	0	0	0	1	0	1	0	0A
Desliga display, mantendo os dados na DDRAM, liga cursor	0	0	0	0	1	0	1	1	0B
Liga o display e esconde o cursor piscante	0	0	0	0	1	1	0	0	0C
Liga o display e o cursor fica piscando	0	0	0	0	1	1	0	1	0D
Liga o display e o cursor fica fixo	0	0	0	0	1	1	1	0	0E
Liga o display e o cursor fica alternante	0	0	0	0	1	1	1	1	0F
Desloca cursor para esquerda e decrementa AC	0	0	0	1	0	0	x	x	10-13
Desloca cursor para direita e incrementa AC	0	0	0	1	0	1	x	x	14-17
Desloca mensagem para esquerda e cursor acompanha	0	0	0	1	1	0	x	x	18-1B
Desloca mensagem para direita e cursor acompanha	0	0	0	1	1	1	x	x	1C-1F
Interface 4 bits, display de 1 linha e matriz 5x8	0	0	1	0	0	0	x	x	20-23
Interface 4 bits, display de 1 linha e matriz 5x11	0	0	1	0	0	1	x	x	24-27
Interface 4 bits, display de 2 linha e matriz 5x8	0	0	1	0	1	0	x	x	28-2B
Interface 4 bits, display de 2 linha e matriz 5x11	0	0	1	0	1	1	x	x	2C-2F
Interface 8 bits, display de 1 linha e matriz 5x8	0	0	1	1	0	0	x	x	30-33
Interface 8 bits, display de 1 linha e matriz 5x11	0	0	1	1	0	1	x	x	34-37
Interface 8 bits, display de 2 linha e matriz 5x8	0	0	1	1	1	0	x	x	38-3B
Interface 8 bits, display de 2 linha e matriz 5x11	0	0	1	1	1	1	x	x	3C-3F
Endereços para escrever na CGRAM	0	1	AC5	AC4	AC3	AC2	AC1	AC0	40-7F
Endereços para escrever/ler na DDRAM	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	80-FF

Tabela 4) código das instruções que comandam o LCD

INSTRUÇÃO	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Descrição	Execução
Limpa Display	0	0	0	0	0	0	0	0	0	1	Limpa todo o display e retoma o cursor para a primeira posição da primeira linha.	1,6 ms
Retorno do cursor	0	0	0	0	0	0	0	0	1	-	Retorna o cursor para a 1ª coluna da 1ª linha. Retorna a mensagem previamente deslocada a sua posição original.	1,6 ms
Fixa o modo de Funcionamento	0	0	0	0	0	0	0	1	X	S	Ajusta o sentido de deslocamento do cursor (X=0 p/ a esquerda, X=1 p/ a direita). Determina se a mensagem deve ou não ser deslocada com a entrada de um novo caractere (S = 1, SIM). Esta instrução tem efeito somente durante a leitura e escrita de dados.	40 µs
Controle do Display	0	0	0	0	0	0	1	D	C	B	Liga (D=1) ou desliga display (D=0). Liga (C=1) ou desliga cursor (C=0). Cursor piscante (B=1) se C=1.	40 µs
Desloca cursor ou mensagem	0	0	0	0	0	1	C	R	-	-	Desloca o cursor (C=0) ou a mensagem (C=1) para a direita se R=1 ou esquerda se R=0. Desloca sem alterar o conteúdo da DDRAM	40 µs
Fixa modo de utilização do módulo LCD	0	0	0	0	1	Y	N	F	-	-	Comunicação do módulo com 8 bits (Y=1) ou 4 bits (Y=0). Número de linhas: 1 (N=0) e 2 ou mais (N=1). Matriz do caractere: 5x7 (F=0) ou 5x10 (F=1). Esta instrução deve ser empregada na inicialização.	40 µs
Endereço da CGRAM	0	0	0	1	Endereço da CGRAM						Fixa o endereço da CGRAM para posterior envio ou leitura de um dado (byte).	40 µs
Endereço da DDRAM	0	0	1	Endereço da DDRAM							Fixa o endereço da DDRAM para posterior envio ou leitura de um dado (byte).	40 µs
Leitura do bit de ocupado e do conteúdo de endereços	0	1	B F	AC							Lê o conteúdo do contador de endereços AC e o BF. O bit 7 do BF indica se a última operação foi concluída (BF=0 concluída, BF=1 em execução).	-
Escreve dado na CGRAM/ DDRAM	1	0	Dado a ser gravado no LCD								Grava o byte presente nos pinos de dados no local apontado pelo contador de endereços (posição do cursor).	40 µs
Lê dado da CGRAM/ DDRAM	1	1	Dado lido do módulo								Lê o byte do local apontado pelo contador de endereços (posição do cursor).	40 µs

Tab. B3: Resumo dos códigos de instruções.

Descrição	Modo	Código Hexa
Controle do display	Liga (sem cursor)	0x0C
	Desliga	0x0A/0x08
Limpa display com retorno do cursor		0x01
Controle do cursor	Liga	0x0E
	Desliga	0x0C
	Desloca p/ a esquerda	0x10
	Desloca p/ a direita	0x14
	Retorno	0x02
	Cursor piscante	0x0D
	Cursor com alternância	0x0F
Sentido de deslocamento do cursor na entrada de um caractere	Para a esquerda	0x04
	Para a direita	0x06
Deslocamento da mensagem na entrada de um caractere	Para a esquerda	0x07
	Para a direita	0x05
Deslocamento da mensagem sem a entrada de caractere	Para a esquerda	0x18
	Para a direita	0x1C
Endereço da primeira posição do cursor	Primeira linha	0x80
	Segunda linha	0xC0

Acionando LCD

- O registrador de instruções (IR - Instruction Register):
- Endereça a memória com os endereços descritos na Tabela 2 e Tabela 3

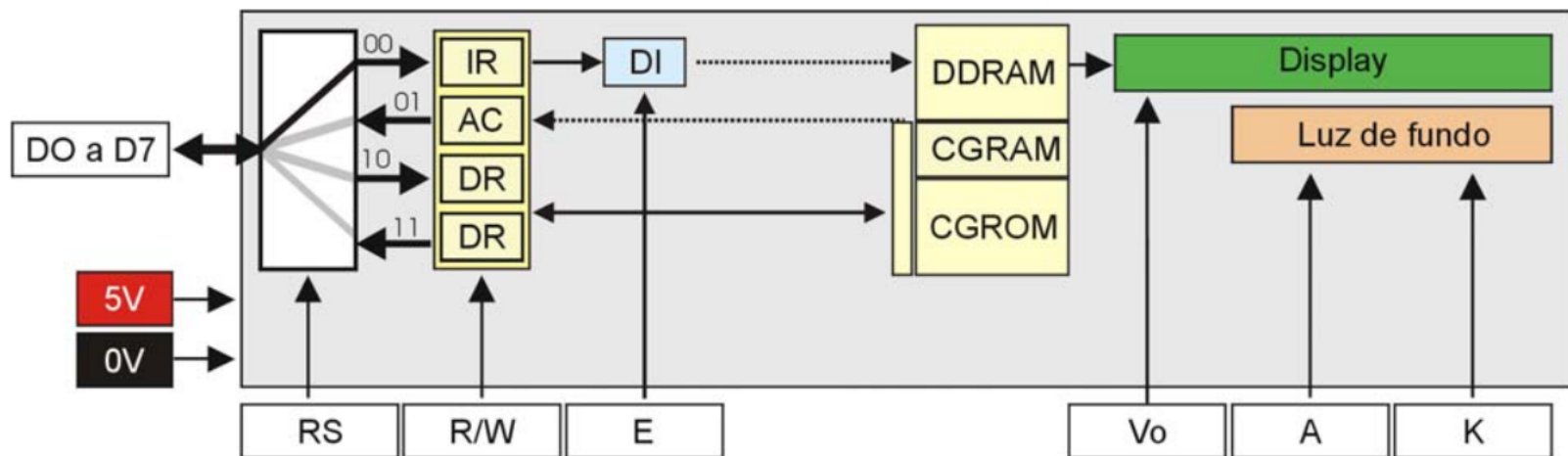


Figura 3) arquitetura interna hipotética (proposta pelo autor)

Acionando LCD

Descrição	Modo	RS	RW	IR
Endereços DDRAM*	Primeira posição da primeira linha	0	0	80
	Primeira posição da segunda linha	0	0	C0

Tabela 2) endereços da DDRAM

ASCII	Endereço (Hexadecimal)	CGRAM
0	40 a 47	
1	48 a 4F	
2	50 a 58	
3	58 a 5F	
4	60 a 67	
5	68 a 6F	
6	70 a 77	
7	78 a 7F	

Tabela 3) endereços da CGRAM

Acionando LCD

- O registrador de dados, DR (Data Register), serve para armazenar os dados que serão escritos no display
- Os dados seguem o padrão ASCII e também podem ser programados pelo usuário através da memória CGRAM
 - RS deve conter 1 lógico e R/W 0 lógico (Write)

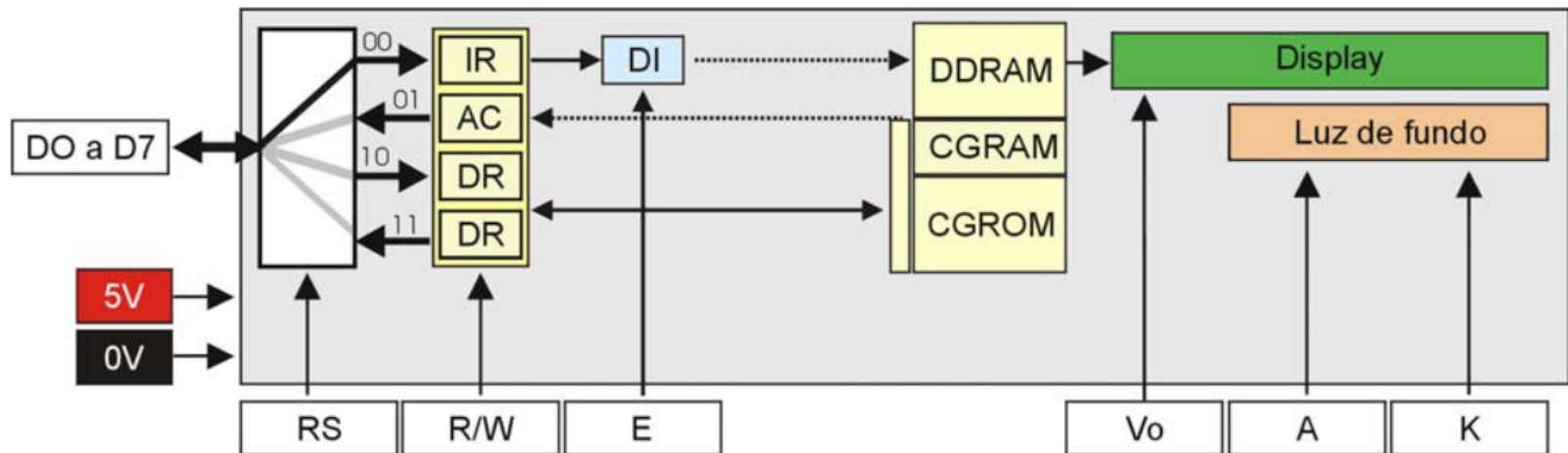


Figura 3) arquitetura interna hipotética (proposta pelo autor)

Acionando LCD

- O registrador AC pode ser incrementado automaticamente quando um caractere é escrito ou decrementado quando uma informação é lida da DDRAM
 - O módulo deve estar programado para tal, de acordo com as instruções descritas na Tabela 4.
 - Esse endereço pode ser lido nos bits D0 a D6 do barramento quando RS=0 e R/W=1
 - O último bit do barramento, D7 conterá o BF (Busy Flag).

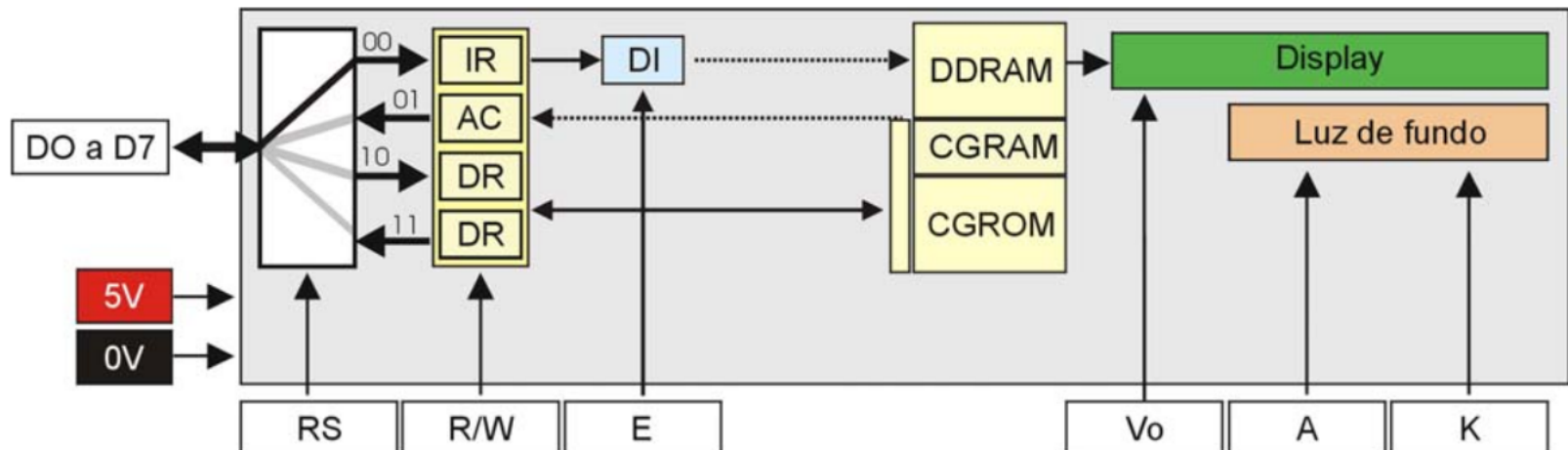
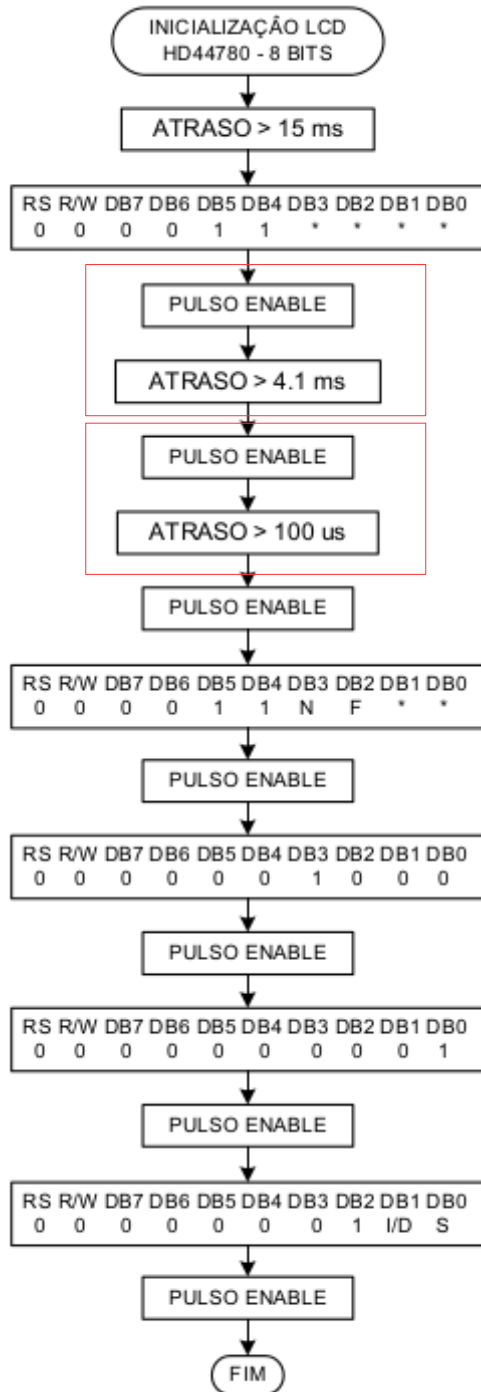


Figura 3) arquitetura interna hipotética (proposta pelo autor)

Modo (Rs=0 e R/W=0)									IR
Limpa o display (DDRAM) e posiciona o cursor no início	0	0	0	0	0	0	0	1	01
Posiciona cursor no início (sem apagar DDRAM)	0	0	0	0	0	0	1	x	02-03
Desloca cursor para esquerda após escrever caractere	0	0	0	0	0	1	0	0	04
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	0	1	05
Desloca cursor para direita após escrever caractere	0	0	0	0	0	1	1	0	06
Desloca mensagem para esquerda após escrever caractere	0	0	0	0	0	1	1	1	07
Desliga display e cursor, mantendo os dados na DDRAM	0	0	0	0	1	0	0	0	08
Desliga display, dados permanecem na DDRAM, cursor pisca	0	0	0	0	1	0	0	1	09
Desliga display e liga cursor fixo	0	0	0	0	1	0	1	0	0A
Desliga display, mantendo os dados na DDRAM, liga cursor	0	0	0	0	1	0	1	1	0B
Liga o display e esconde o cursor piscante	0	0	0	0	1	1	0	0	0C
Liga o display e o cursor fica piscando	0	0	0	0	1	1	0	1	0D
Liga o display e o cursor fica fixo	0	0	0	0	1	1	1	0	0E
Liga o display e o cursor fica alternante	0	0	0	0	1	1	1	1	0F
Desloca cursor para esquerda e decrementa AC	0	0	0	1	0	0	x	x	10-13
Desloca cursor para direita e incrementa AC	0	0	0	1	0	1	x	x	14-17
Desloca mensagem para esquerda e cursor acompanha	0	0	0	1	1	0	x	x	18-1B
Desloca mensagem para direita e cursor acompanha	0	0	0	1	1	1	x	x	1C-1F
Interface 4 bits, display de 1 linha e matriz 5x8	0	0	1	0	0	0	x	x	20-23
Interface 4 bits, display de 1 linha e matriz 5x11	0	0	1	0	0	1	x	x	24-27
Interface 4 bits, display de 2 linha e matriz 5x8	0	0	1	0	1	0	x	x	28-2B
Interface 4 bits, display de 2 linha e matriz 5x11	0	0	1	0	1	1	x	x	2C-2F
Interface 8 bits, display de 1 linha e matriz 5x8	0	0	1	1	0	0	x	x	30-33
Interface 8 bits, display de 1 linha e matriz 5x11	0	0	1	1	0	1	x	x	34-37
Interface 8 bits, display de 2 linha e matriz 5x8	0	0	1	1	1	0	x	x	38-3B
Interface 8 bits, display de 2 linha e matriz 5x11	0	0	1	1	1	1	x	x	3C-3F
Endereços para escrever na CGRAM	0	1	AC5	AC4	AC3	AC2	AC1	AC0	40-7F
Endereços para escrever/ler na DDRAM	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	80-FF

Tabela 4) código das instruções que comandam o LCD



Esperar mais de 15 ms após VCC alcançar 4,5 V

Interface de 8 bits

Fixa modo de utilização do módulo LCD	0	0	0	0	1	Y	N	F	-	-	Comunicação do módulo com 8 bits (Y=1) ou 4 bits (Y=0). Número de linhas: 1 (N=0) e 2 ou mais (N=1). Matriz do caractere: 5×7 (F=0) ou 5×10 (F=1). Esta instrução deve ser empregada na inicialização.	40 s
---------------------------------------	---	---	---	---	---	---	---	---	---	---	---	------

Ajusta o modo de utilização do display

Desliga o display

Controle do Display	0	0	0	0	0	0	1	D	C	B	Liga (D=1) ou desliga display (D=0). Liga (C=1) ou desliga cursor (C=0). Cursor piscante (B=1) se C=1.	40 s
---------------------	---	---	---	---	---	---	---	---	---	---	--	------

Limpa o display

Limpa Display	0	0	0	0	0	0	0	0	0	1	Limpa todo o display e retorna o cursor para a primeira posição da primeira linha.	1,6 ms
---------------	---	---	---	---	---	---	---	---	---	---	--	--------

Ajusta o modo de funcionamento do cursor

Fixa o modo de Funcionamento	0	0	0	0	0	0	0	1	X	S	Ajusta o sentido de deslocamento do cursor (X=0 p/ a esquerda, X=1 p/ a direita). Determina se a mensagem deve ou não ser deslocada com a entrada de um novo caractere (S = 1, SIM). Esta instrução tem efeito somente durante a leitura e escrita de dados.	40 s
------------------------------	---	---	---	---	---	---	---	---	---	---	--	------

Acionando LCD

- **Existem duas possibilidades de comunicação com o display**
 - **Empregando 8 vias de dados para a comunicação (D0-D7)**
 - **Empregando 4 vias de dados (D4-D7)**
 - O dado é enviado separadamente em duas partes (2 nibbles)

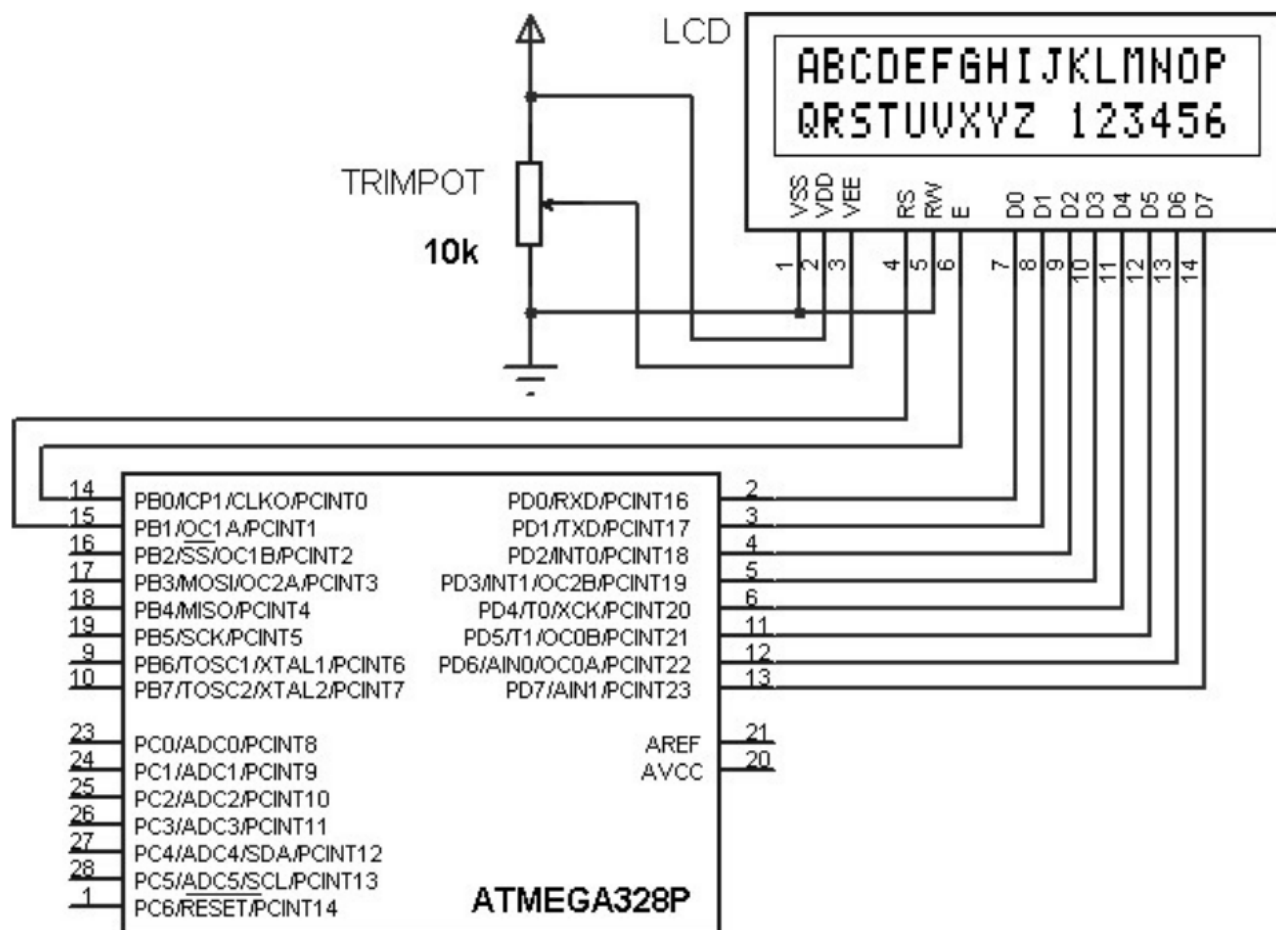


Fig. 5.16 – Circuito para acionamento de um LCD 16 × 2 usando 8 vias de dados.

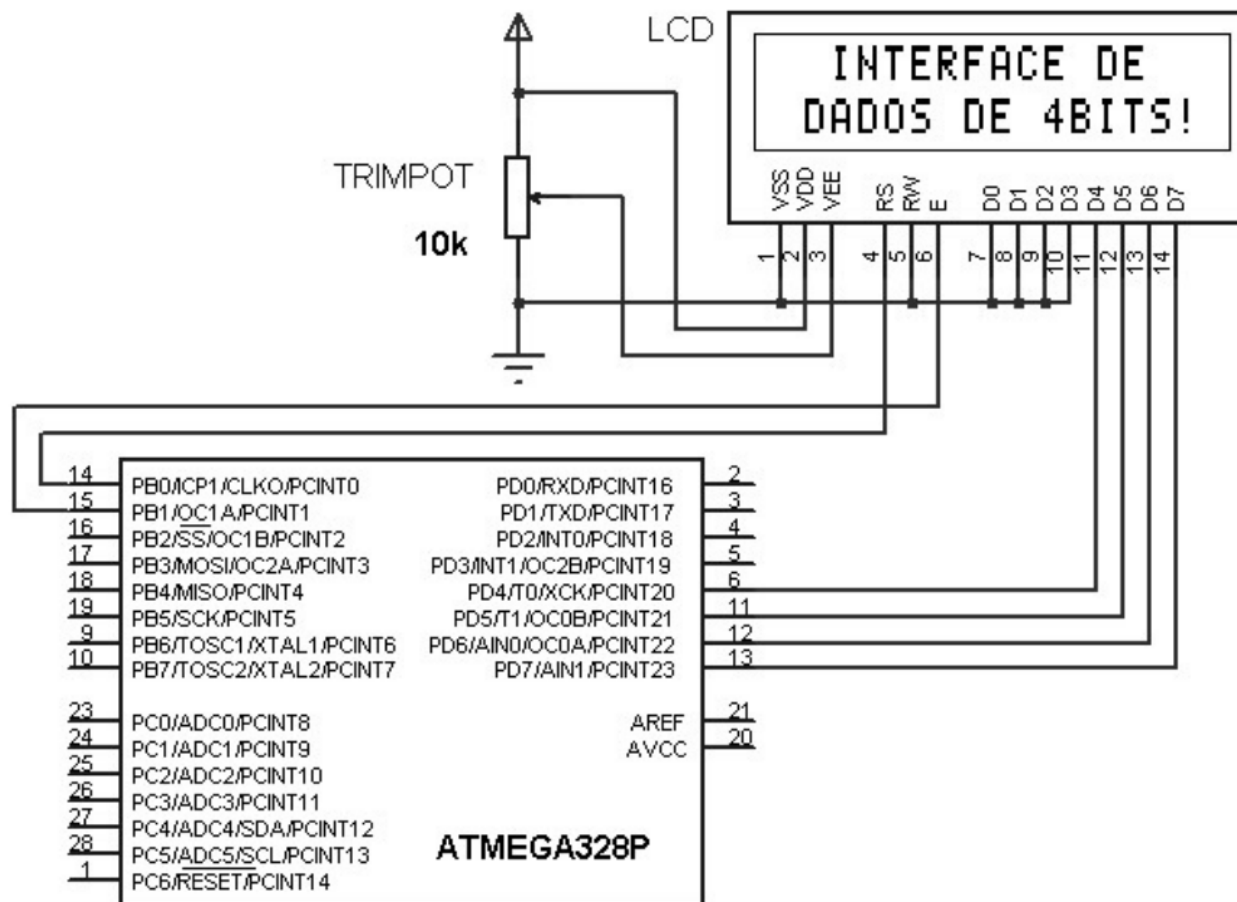


Fig. 5.18 – Circuito para acionamento de um LCD 16 × 2 com interface de dados de 4 bits.

Acionando LCD

- **Interface de dados de 8 bits**
- **Para ler ou escrever no display LCD:**
 - 1. Levar o pino R/W para:
 - » **0 lógico - para operação de escrita (Write)**
 - » **1 lógico - para operação de leitura (Read)**
 - » **Aterra-se (GND) esse pino se não há necessidade de monitorar a resposta do LCD – sempre 0 lógico (Write).**
 - 2. Levar o pino RS (Register Select) para:
 - » **0 lógico - instrução**
 - » **1 lógico - caractere**

Acionando LCD

- **Interface de dados de 8 bits**
- **Para ler ou escrever no display LCD:**
 - 3. Transferir os dados para a via de dados (8 bits)
 - 4. Gerar um pulso de habilitação
 - » **Levar o pino E (Enable) para 1 lógico e, após um pequeno tempo, para 0 lógico**
 - 5. Empregar uma rotina de atraso entre as instruções ou
 - » **Fazer a leitura do busy flag (o bit 7 da linha de dados que indica que o display está ocupado) antes do envio da instrução, enviando-a somente quando esse flag for 0 lógico.**
 - » **Utilizaremos uma rotina de atraso entre as instruções**

Acionando LCD

- **Interface de dados de 8 bits**
- **Os passos 1, 2 e 3 podem ser efetuados em qualquer sequência**
 - O pulso de habilitação é que faz o controlador do LCD ler os dados dos seus pinos
 - É importante respeitar os tempos de resposta do LCD à transição dos sinais enviados ao mesmo
- **O fluxograma de inicialização do LCD conforme especificação da Hitachi**
 - Se desejado, o busy flag pode ser lido após o ajuste do modo de utilização do display
 - Os comandos para o LCD são detalhado no apêndice B.

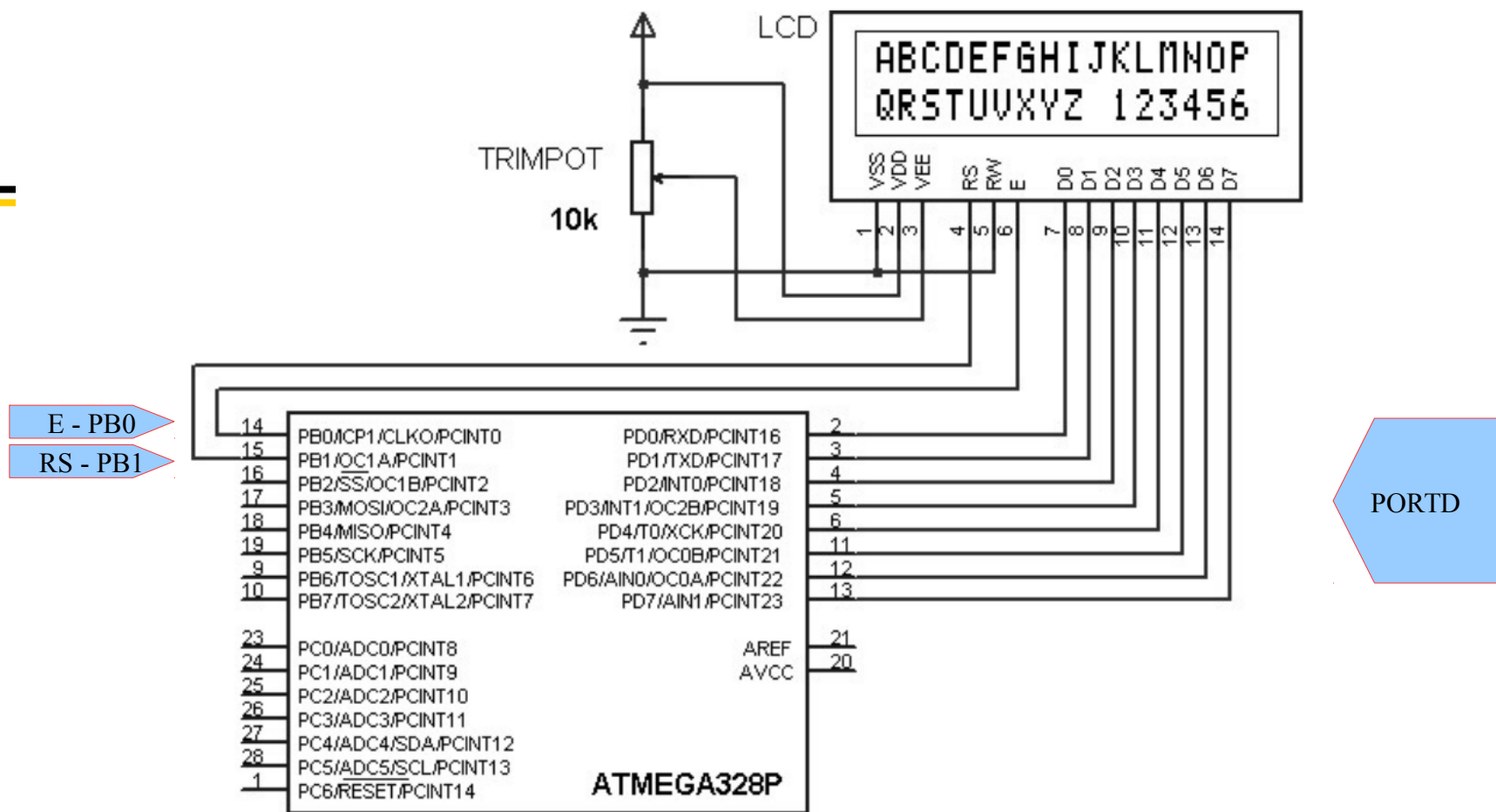
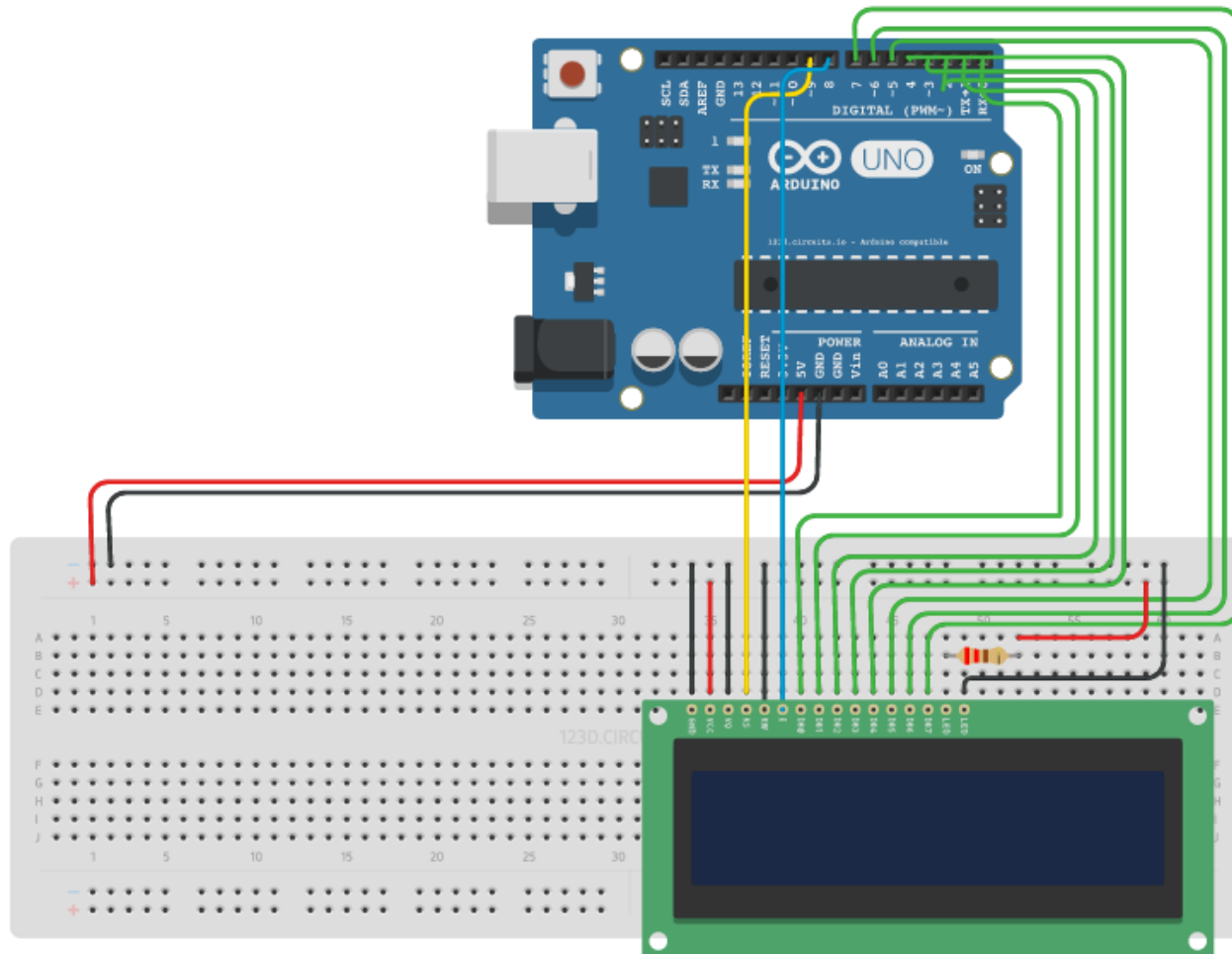


Fig. 5.16 – Circuito para acionamento de um LCD 16 × 2 usando 8 vias de dados.

```
#define DADOS_LCD    PORTD    //8 bits de dados do LCD na porta D
#define CONTR_LCD    PORTB    //os pino de controle estão no PORTB
#define RS           PB1      //pino de instrução ou dado para o LCD
#define E            PB0      //pino de enable do LCD
```

<https://www.tinkercad.com/things/19Wb43hH170>



```

//----- //
//   AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012. //
//----- //
//===== //
//   ACIONANDO UM DISPLAY DE CRISTAL LIQUIDO DE 16x2 //
//                                     //
//   Interface de dados de 8 bits //
//===== //
#define F_CPU 16000000UL //define a frequência do microcontrolador - 16MHz

#include <avr/io.h> //definições do componente especificado
#include <util/delay.h> //biblioteca para o uso das rotinas de delay
#include <avr/pgmspace.h> //uso de funções para salvar dados na memória de programa

//Definições de macros - empregadas para o trabalho com o bits
#define set_bit(Y,bit_x) (Y|=(1<<bit_x)) //ativa o bit x da variável Y
#define clr_bit(Y,bit_x) (Y&=~(1<<bit_x)) //limpa o bit x da variável Y
#define tst_bit(Y,bit_x) (Y&(1<<bit_x)) //testa o bit x da variável Y
#define cpl_bit(Y,bit_x) (Y^=(1<<bit_x)) //troca o estado do bit x da variável Y

//para uso no LCD
#define pulso_enable() _delay_us(1); set_bit(CONTR_LCD,E); _delay_us(1); clr_bit(CONTR_LCD,E); _delay_us(45)

#define DADOS_LCD PORTD //8 bits de dados do LCD na porta D
#define CONTR_LCD PORTB //os pinos de controle estão no PORTB
#define RS PB1 //pino de instrução ou dado para o LCD
#define E PB0 //pino de enable do LCD

//mensagem armazenada na memória flash
const unsigned char msg1[] PROGMEM = "ABCDEFGHJKLMNOP";

```

```

//-----
int main()
{
    unsigned char i;

    DDRB = 0xFF;      //PORTB como saída
    DDRD = 0xFF;      //PORTD como saída
    UCSRB = 0x00;     //habilita os pinos PD0 e PD1 como I/O para uso no Arduino

    inic_LCD_8bits(); //inicializa o LCD

    for(i=0;i<16;i++) //enviando caractere por caractere
        cmd_LCD(pgm_read_byte(&msg1[i]),1); //lê na memória flash e usa cmd_LCD

    cmd_LCD(0xC0,0);  //desloca o cursor para a segunda linha do LCD
    escreve_LCD("QRSTUVWXYZ 123456"); //a cadeia de caracteres é criada na RAM

    for(;;);          //laço infinito
}
//=====

```

```

//-----
//Sub-rotina de inicialização do LCD - sequência ditada pelo fabricante do circuito de controle do LCD
//-----
void inic_LCD_8bits()
{
    clr_bit(CONTR_LCD,RS); //o LCD será só escrito então R/W é sempre zero

    _delay_ms(15);        /*tempo para estabilizar a tensão do LCD, após VCC ultrapassar
                           4.5 V (pode ser bem maior na prática)*/

    DADOS_LCD = 0x38; //interface 8 bits, 2 linhas, matriz 7x5 pontos

    pulso_enable();        //enable respeitando os tempos de resposta do LCD
    _delay_ms(5);
    pulso_enable();
    _delay_us(200);
    pulso_enable();
    pulso_enable();

    cmd_LCD(0x08,0); //desliga LCD
    cmd_LCD(0x01,0); //limpa todo o display
    cmd_LCD(0x0C,0); //mensagem aparente cursor inativo não piscando
    cmd_LCD(0x80,0); //escreve na primeira posição a esquerda - 1ª linha
}
//

```

```

//-----
//Sub-rotina de inicialização do LCD -
//-----
void inic_LCD_8bits()
{
    clr_bit(CONTR_LCD,RS); //o LCD será :
    _delay_ms(15);        /*tempo para estabilizar a tensão em 4.5 V (pode ser bem mais)*/

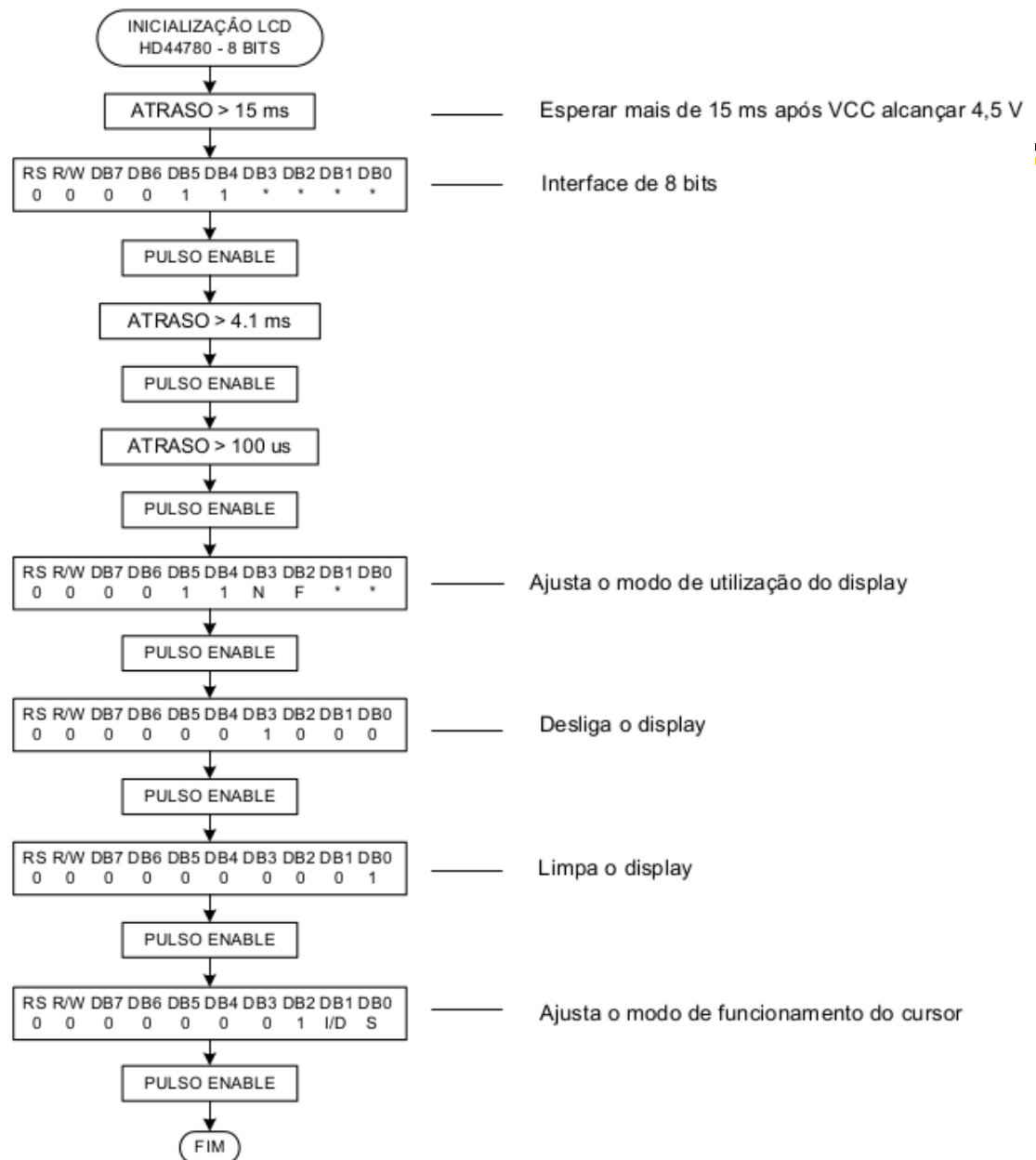
    DADOS_LCD = 0x38; //interface 8 bits;

    pulso_enable();        //enable respectivamente
    _delay_ms(5);
    pulso_enable();
    _delay_us(200);
    pulso_enable();
    pulso_enable();

    cmd_LCD(0x08,0); //desliga LCD
    cmd_LCD(0x01,0); //limpa todo o display
    cmd_LCD(0x0C,0); //mensagem aparece no display
    cmd_LCD(0x80,0); //escreve na primeira linha

}
//

```



```

//-----
//Sub-rotina de escrita no LCD
//-----
void escreve_LCD(char *c)
{
    for (; *c!=0;c++) cmd_LCD(*c,1);
}
//-----

//-----
//Sub-rotina para enviar caracteres e comandos ao LCD
//-----
void cmd_LCD(unsigned char c, char cd) //c é o dado e cd indica se é instrução ou caractere
{
    DADOS_LCD = c;

    if(cd==0)
        clr_bit(CONTR_LCD,RS); //RS = 0
    else
        set_bit(CONTR_LCD,RS); //RS = 1

    pulso_enable();

    //se for instrução de limpeza ou retorno de cursor espera o tempo necessário
    if((cd==0) && (c<4))
        _delay_ms(2);
}
//-----

```

Acionando LCD

- **Interface de dados de 4 bits**
- **Para ler ou escrever no display LCD:**
 - 1. Levar o pino R/W para:
 - » 0 lógico - para operação de escrita (Write)
 - » 1 lógico - para operação de leitura (Read)
 - » **Aterra-se (GND) esse pino se não há necessidade de monitorar a resposta do LCD – sempre 0 lógico (Write).**
 - 2. Levar o pino RS (Register Select) para:
 - » 0 lógico - instrução
 - » 1 lógico - caractere

Acionando LCD

- Interface de dados de 4 bits
- Para ler ou escrever no display LCD:
 - 3. Transferir a parte mais significativa dos dados para a via de dados
 - » 4 bits mais significativos (MSB) – *nibble* maior
 - 4. Gerar um pulso de habilitação
 - » Levar o pino E (Enable) para 1 lógico e, após um pequeno tempo, para 0 lógico
 - 5. Transferir a parte menos significativa dos dados para a via de dados
 - » 4 bits menos significativos (LSB) – *nibble* menor
 - 6. Gerar outro pulso de habilitação

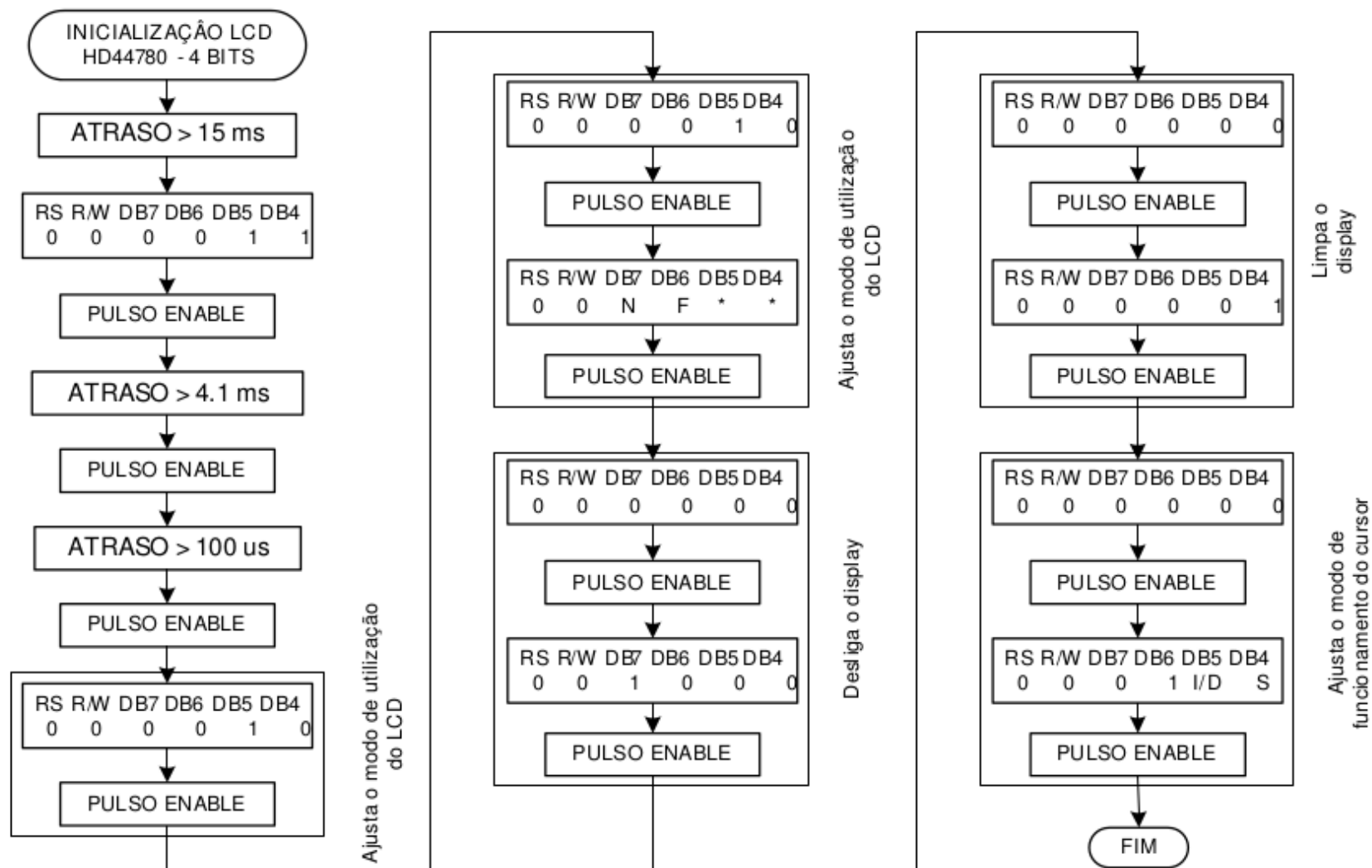


Fig. 5.19 – Rotina de inicialização de 4 bits para um LCD com base no CI HD44780.

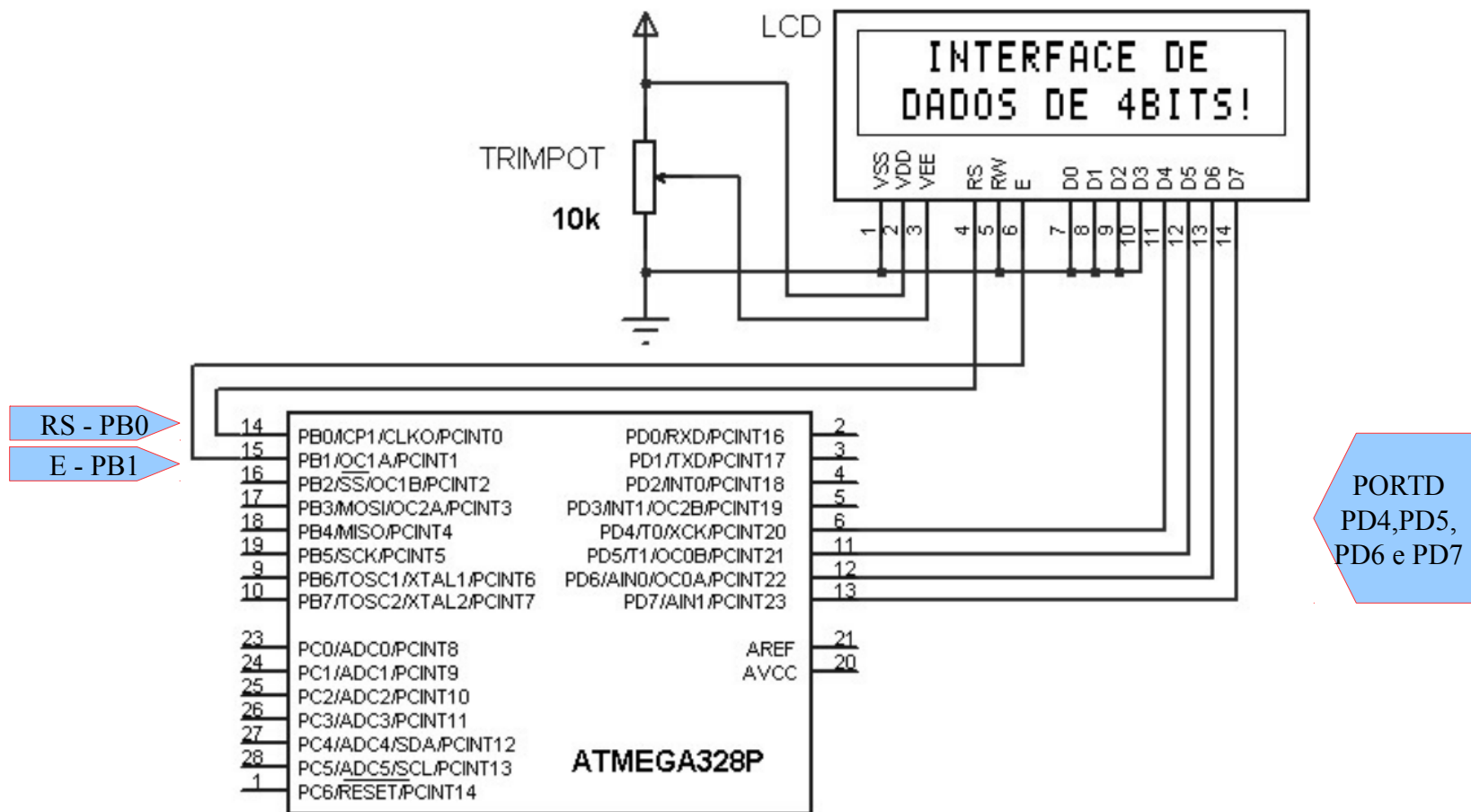


Fig. 5.18 – Circuito para acionamento de um LCD 16 × 2 com interface de dados de 4 bits.

def_principais.h

```
#ifndef _DEF_PRINCIPAIS_H
#define _DEF_PRINCIPAIS_H

#define F_CPU 16000000UL    //define a frequencia do microcontrolador - 16MHz

#include <avr/io.h>          //definições do componente especificado
#include <util/delay.h>       //biblioteca para o uso das rotinas de _delay_ms e _delay_us()
#include <avr/pgmspace.h>     //para o uso do PROGMEM, gravação de dados na memória flash

//Definições de macros para o trabalho com bits

#define set_bit(y,bit) (y|=(1<<bit)) //coloca em 1 o bit x da variável Y
#define clr_bit(y,bit) (y&=~(1<<bit)) //coloca em 0 o bit x da variável Y
#define cpl_bit(y,bit) (y^=(1<<bit)) //troca o estado lógico do bit x da variável Y
#define tst_bit(y,bit) (y&(1<<bit))  //retorna 0 ou 1 conforme leitura do bit

#endif
```

LCD.h

```
#ifndef _LCD_H
#define _LCD_H

#include "def_principais.h"

//Definições para facilitar a troca dos pinos do hardware e facilitar a re-programação

#define DADOS_LCD      PORTD      //4 bits de dados do LCD no PORTD
#define nibble_dados 1    //0 para via de dados do LCD nos 4 LSBs do PORT empregado (Px0-D4, Px1-D5, Px2-D6, Px3-D7)
                        //1 para via de dados do LCD nos 4 MSBs do PORT empregado (Px4-D4, Px5-D5, Px6-D6, Px7-D7)
#define CONTR_LCD      PORTE      //PORT com os pinos de controle do LCD (pino R/W em 0).
#define E              PB1        //pino de habilitação do LCD (enable)
#define RS              PB0        //pino para informar se o dado é uma instrução ou caractere

#define tam_vetor 5    //número de dígitos individuais para a conversão por ident_num()
#define conv_ascii 48 //48 se ident_num() deve retornar um número no formato ASCII (0 para formato normal)

//sinal de habilitação para o LCD
#define pulso_enable() _delay_us(1); set_bit(CONTR_LCD,E); _delay_us(1); clr_bit(CONTR_LCD,E); _delay_us(45)

//protótipo das funções
void cmd_LCD(unsigned char c, char cd);
void inic_LCD_4bits();
void escreve_LCD(char *c);
void escreve_LCD_Flash(const char *c);

void ident_num(unsigned int valor, unsigned char *disp);

#endif
```

LCD_4bits.c

```
//-----  
//    AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012.    //  
//-----  
//===== //  
//    ACIONANDO UM DISPLAY DE CRISTAL LIQUIDO DE 16x2    //  
//    Interface de dados de 4 bits    //  
//===== //  
  
#include "def_principais.h"    //inclusão do arquivo com as principais definições  
#include "LCD.h"  
  
//definição para acessar a memória flash  
PROGMEM const char mensagem[] = " DADOS DE 4BITS!\0"; //mensagem armazenada na memória flash  
  
//-----  
int main()  
{  
    DDRD = 0xFF;    //PORTD como saída  
    DDRB = 0xFF;  
  
    inic_LCD_4bits();    //inicializa o LCD  
    escreve_LCD(" INTERFACE DE");    //string armazenada na RAM  
    cmd_LCD(0xC0,0);    //desloca cursor para a segunda linha  
    escreve_LCD_Flash(mensagem);    //string armazenada na flash  
  
    for(;;){}    //laço infinito  
}  
//=====
```

Acionando LCD

- A função `inic_LCD_4bits()` deve ser utilizada no início do programa principal para a correta inicialização do LCD
- Existe uma sequência de comandos que deve ser seguida para que o LCD possa funcionar corretamente

LCD.c

```
//-----  
//Sub-rotina para inicialização do LCD com via de dados de 4 bits  
//-----  
void inic_LCD_4bits()      //sequência ditada pelo fabricante do circuito integrado HD44780  
{                          //o LCD será só escrito. Então, R/W é sempre zero.  
  
    clr_bit(CONTR_LCD,RS); //RS em zero indicando que o dado para o LCD será uma instrução  
    clr_bit(CONTR_LCD,E);  //pino de habilitação em zero  
  
    _delay_ms(20);         //tempo para estabilizar a tensão do LCD, após VCC ultrapassar 4.5 V (na prática pode  
                           //ser maior).  
    //interface de 8 bits  
    #if (nibble_dados)  
        DADOS_LCD = (DADOS_LCD & 0x0F) | 0x30;  
    #else  
        DADOS_LCD = (DADOS_LCD & 0xF0) | 0x03;  
    #endif  
  
    pulso_enable();        //habilitação respeitando os tempos de resposta do LCD  
    _delay_ms(5);  
    pulso_enable();  
    _delay_us(200);  
    pulso_enable();        /*até aqui ainda é uma interface de 8 bits.  
                           Muitos programadores desprezam os comandos acima, respeitando apenas o tempo de  
                           estabilização da tensão (geralmente funciona). Se o LCD não for inicializado primeiro no  
                           modo de 8 bits, haverá problemas se o microcontrolador for inicializado e o display já o tiver sido.*/  
  
    //interface de 4 bits, deve ser enviado duas vezes (a outra está abaixo)  
    #if (nibble_dados)  
        DADOS_LCD = (DADOS_LCD & 0x0F) | 0x20;  
    #else  
        DADOS_LCD = (DADOS_LCD & 0xF0) | 0x02;  
    #endif  
  
    pulso_enable();  
    cmd_LCD(0x28,0);        //interface de 4 bits 2 linhas (aqui se habilita as 2 linhas)  
                           //são enviados os 2 nibbles (0x2 e 0x8)  
    cmd_LCD(0x08,0);        //desliga o display  
    cmd_LCD(0x01,0);        //limpa todo o display  
    cmd_LCD(0x0C,0);        //mensagem aparente cursor inativo não piscando  
    cmd_LCD(0x80,0);        //inicializa cursor na primeira posição a esquerda - 1a linha  
}
```

Acionando LCD

- **A função `escreve_LCD("frase")` recebe uma string, ou seja um conjunto de caracteres**
- **Como na programação em C toda string é finalizada com o caractere nulo (0), essa função se vale desse artifício para verificar o final da string**
 - Deve-se ter cuidado ao se utilizar essa função, porque a string é armazenada na memória RAM do microcontrolador, o que pode limitar a memória disponível para o programa.
- **A função `escreve_LCD_Flash(frase)`, onde a frase, previamente declarada no programa, é armazenada na memória flash**

LCD.c

```
//-----  
//Sub-rotina de escrita no LCD - dados armazenados na RAM  
//-----  
void escreve_LCD(char *c)  
{  
    for (; *c!=0;c++) cmd_LCD(*c,1);  
}  
//-----  
//Sub-rotina de escrita no LCD - dados armazenados na FLASH  
//-----  
void escreve_LCD_Flash(const char *c)  
{  
    for (;pgm_read_byte(&(*c))!=0;c++) cmd_LCD(pgm_read_byte(&(*c)),1);  
}  
//-----  
//Conversão de um número em seus dígitos individuais  
//-----  
void ident_num(unsigned int valor, unsigned char *disp)  
{  
    unsigned char n;  
  
    for(n=0; n<tam_vetor; n++)  
        disp[n] = 0 + conv_ascii;    //limpa vetor para armazenagem do dígitos  
  
    do  
    {  
        *disp = (valor%10) + conv_ascii;    //pega o resto da divisão por 10  
        valor /=10;    //pega o inteiro da divisão por 10  
        disp++;  
    }while (valor!=0);  
}  
//-----
```

Acionando LCD

- A principal função para o controle do LCD é a **cmd_LCD(dado, 0 ou 1)**
- Recebe dois parâmetros:
 - o dado que se deseja enviar ao LCD e
 - o número 0 ou 1
 - » 0 indica que o dado é uma instrução
 - » 1 indica que o dado é um caractere

LCD.c

```
#include "LCD.h"

//-----
// Sub-rotina para enviar caracteres e comandos ao LCD com via de dados de 4 bits
//-----
void cmd_LCD(unsigned char c, char cd)          //c é o dado e cd indica se é instrução ou caractere
{
    if(cd==0)
        clr_bit(CONTR_LCD,RS);
    else
        set_bit(CONTR_LCD,RS);

    //primeiro nibble de dados - 4 MSB
    #if (nibble_dados)                          //compila código para os pinos de dados do LCD nos 4 MSB do PORT
        DADOS_LCD = (DADOS_LCD & 0x0F) | (0xF0 & c);
    #else                                       //compila código para os pinos de dados do LCD nos 4 LSB do PORT
        DADOS_LCD = (DADOS_LCD & 0xF0) | (c >> 4);
    #endif

    pulso_enable();

    //segundo nibble de dados - 4 LSB
    #if (nibble_dados)                          //compila código para os pinos de dados do LCD nos 4 MSB do PORT
        DADOS_LCD = (DADOS_LCD & 0x0F) | (0xF0 & (c << 4));
    #else                                       //compila código para os pinos de dados do LCD nos 4 LSB do PORT
        DADOS_LCD = (DADOS_LCD & 0xF0) | (0x0F & c);
    #endif

    pulso_enable();

    if((cd==0) && (c<4))                      //se for instrução de retorno ou limpeza espera LCD estar pronto
        _delay_ms(2);
}
```

Novos Caracteres

- **CGRAM (Caractere Generator RAM)**
- Cada matriz utiliza 8 bytes totalizando uma memória de 64 bytes acessados através de um conjunto de endereços

ASCII	Endereço (Hexadecimal)	CGRAM
0	40 a 47	
1	48 a 4F	
2	50 a 58	
3	58 a 5F	
4	60 a 67	
5	68 a 6F	
6	70 a 77	
7	78 a 7F	

Tabela 3) endereços da CGRAM

Tab. 5.3 – Endereço para criação de caracteres novos e seus códigos de chamada.

Endereço base	0x40	0x48	0x50	0x58	0x60	0x68	0x70	0x78
Código do caractere	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07

Novos Caracteres

- **CGRAM (Caractere Generator RAM)**
- Os dados da CGRAM são apresentados em um mapa de bits de 8 bytes
 - Utilizam 7, com 5 bits cada, sendo 1 byte reservado para o cursor

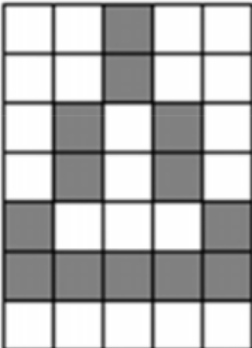
Endereço da CGRAM	Mapa de bits	Dado
0x48		0b00100
0x49		0b00100
0x4A		0b01010
0x4B		0b01010
0x4C		0b10001
0x4E		0b11111
0x4F		0b00000

Fig. 5.24 – Gravação do símbolo Δ na CGRAM, matriz 5×7 . Esse caractere será selecionado pelo código 0x01.

LCD_4bits_2new_caract.c

```
//-----  
//    AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012.    //  
//-----  
//===== //  
//          CRIANDO CARACTERES PARA O LCD 16x2          //  
//                                     //  
//          Via de dados de 4 bits                                     //  
//===== //  
  
#include "def_principais.h"          //inclusão do arquivo com as principais definições  
#include "LCD.h"  
  
//informação para criar caracteres novos armazenada na memória flash  
const unsigned char carac1[] PROGMEM = {0b01110, //Ç  
                                          0b10001,  
                                          0b10000,  
                                          0b10000,  
                                          0b10101,  
                                          0b01110,  
                                          0b10000};  
  
const unsigned char carac2[] PROGMEM = {0b00100, //Delta  
                                          0b00100,  
                                          0b01010,  
                                          0b01010,  
                                          0b10001,  
                                          0b11111,  
                                          0b00000};  
  
//-----
```


LCD_4bits_2new_caract.c

```
//-----  
int main()  
{  
    unsigned char k;  
  
    DDRD = 0xFF;           //PORTD como saída  
    DDRB = 0xFF;           //PORTB como saída  
  
    inic_LCD_4bits();       //inicializa o LCD  
  
    cmd_LCD(0x40,0);        //endereço base para gravar novo segmento 0x40  
    for(k=0;k<7;k++)  
        cmd_LCD(pgm_read_byte(&carac1[k]),1); //grava 8 bytes na DDRAM começando no end. 0x40  
    cmd_LCD(0x00,1);        //apaga última posição do end. da CGRAM para evitar algum dado espúrio  
  
    cmd_LCD(0x48,0);        //endereço base para gravar novo segmento 0x48  
    for(k=0;k<7;k++)  
        cmd_LCD(pgm_read_byte(&carac2[k]),1); //grava 8 bytes na DDRAM começando no end. 0x48  
    cmd_LCD(0x00,1);        //apaga última posição do end. da CGRAM para evitar algum dado espúrio  
  
    cmd_LCD(0x80,0);        //endereça a posição para escrita dos caracteres  
    cmd_LCD(0x00,1);        //apresenta primeiro caractere 0x00  
    cmd_LCD(0x01,1);        //apresenta segundo caractere 0x01  
  
    for(;;); //laço infinito  
}  
//=====
```

Rotinas Decodificação

- **Na programação em C, as variáveis podem ser de 8, 16 ou mais bits**
- Para apresentar em um display o valor de alguma dessas variáveis
- É necessário decodificar o número representado pela variável em seus dígitos individuais:
 - a) Dividir o número por 10
 - b) Guardar o resto
 - c) Pegar o número inteiro resultante
 - d) E seguir com o processo até que a divisão resulte zero
 - e) Os restos da divisão são os n dígitos individuais do número (base decimal)

LCD.h

```
#ifndef _LCD_H
#define _LCD_H

#include "def_principais.h"

//Definições para facilitar a troca dos pinos do hardware e facilitar a re-programação

#define DADOS_LCD      PORTD      //4 bits de dados do LCD no PORTD
#define nibble_dados 1    //0 para via de dados do LCD nos 4 LSBs do PORT empregado (Px0-D4, Px1-D5, Px2-D6, Px3-D7)
                        //1 para via de dados do LCD nos 4 MSBs do PORT empregado (Px4-D4, Px5-D5, Px6-D6, Px7-D7)
#define CONTR_LCD      PORTE      //PORT com os pinos de controle do LCD (pino R/W em 0).
#define E              PB1        //pino de habilitação do LCD (enable)
#define RS              PB0        //pino para informar se o dado é uma instrução ou caractere

#define tam_vetor 5 //número de dígitos individuais para a conversão por ident_num()
#define conv_ascii 48 //48 se ident_num() deve retornar um número no formato ASCII (0 para formato normal)

//sinal de habilitação para o LCD
#define pulso_enable() _delay_us(1); set_bit(CONTR_LCD,E); _delay_us(1); clr_bit(CONTR_LCD,E); _delay_us(45)

//protótipo das funções
void cmd_LCD(unsigned char c, char cd);
void inic_LCD_4bits();
void escreve_LCD(char *c);
void escreve_LCD_Flash(const char *c);

void ident_num(unsigned int valor, unsigned char *disp);

#endif
```

LCD.c

```
//-----  
//Sub-rotina de escrita no LCD - dados armazenados na RAM  
//-----  
void escreve_LCD(char *c)  
{  
    for (; *c!=0;c++) cmd_LCD(*c,1);  
}  
//-----  
//Sub-rotina de escrita no LCD - dados armazenados na FLASH  
//-----  
void escreve_LCD_Flash(const char *c)  
{  
    for (;pgm_read_byte(&(*c))!=0;c++) cmd_LCD(pgm_read_byte(&(*c)),1);  
}  
//-----  
//Conversão de um número em seus dígitos individuais  
//-----  
void ident_num(unsigned int valor, unsigned char *disp)  
{  
    unsigned char n;  
  
    for(n=0; n<tam_vetor; n++)  
        disp[n] = 0 + conv_ascii;    //limpa vetor para armazenagem do dígitos  
  
    do  
    {  
        *disp = (valor%10) + conv_ascii;    //pega o resto da divisão por 10  
        valor /=10;                          //pega o inteiro da divisão por 10  
        disp++;  
    }while (valor!=0);  
}  
//-----
```

Rotinas Decodificação

- **A função recebe dois parâmetros:**
 - Um deles é o valor a ser convertido, limitado ao tamanho da variável declarada, no caso acima, 16 bits (unsigned int);
 - O outro parâmetro é o ponteiro para o vetor (*disp) que conterà os valores individuais dos dígitos da conversão
- **No início da função, o vetor é zerado para garantir que valores anteriormente convertidos não prejudiquem a conversão atual**
- Esse vetor deve ser declarado no corpo do programa onde será utilizado e seu tamanho é determinado pelo máximo valor que poderá conter

LCD.h

```
#ifndef _LCD_H
#define _LCD_H

#include "def_principais.h"

//Definições para facilitar a troca dos pinos do hardware e facilitar a re-programação

#define DADOS_LCD      PORTD      //4 bits de dados do LCD no PORTD
#define nibble_dados 1    //0 para via de dados do LCD nos 4 LSBs do PORT empregado (Px0-D4, Px1-D5, Px2-D6, Px3-D7)
                        //1 para via de dados do LCD nos 4 MSBs do PORT empregado (Px4-D4, Px5-D5, Px6-D6, Px7-D7)
#define CONTR_LCD      PORTE      //PORT com os pinos de controle do LCD (pino R/W em 0).
#define E               PB1       //pino de habilitação do LCD (enable)
#define RS              PB0       //pino para informar se o dado é uma instrução ou caractere

#define tam_vetor 5    //número de dígitos individuais para a conversão por ident_num()
#define conv_ascii 48  //48 se ident_num() deve retornar um número no formato ASCII (0 para formato normal)

//sinal de habilitação para o LCD
#define pulso_enable() _delay_us(1); set_bit(CONTR_LCD,E); _delay_us(1); clr_bit(CONTR_LCD,E); _delay_us(45)

//protótipo das funções
void cmd_LCD(unsigned char c, char cd);
void inic_LCD_4bits();
void escreve_LCD(char *c);
void escreve_LCD_Flash(const char *c);

void ident_num(unsigned int valor, unsigned char *disp);

#endif
```

Rotinas Decodificação

- **Exemplo:**

- Um número de base decimal necessita ser somado a 48 (0x30)
- Para apresentar o número 5 em uma posição do LCD, deve ser enviado o número 53
 - » $0x30 + 0x05 = 0x35$

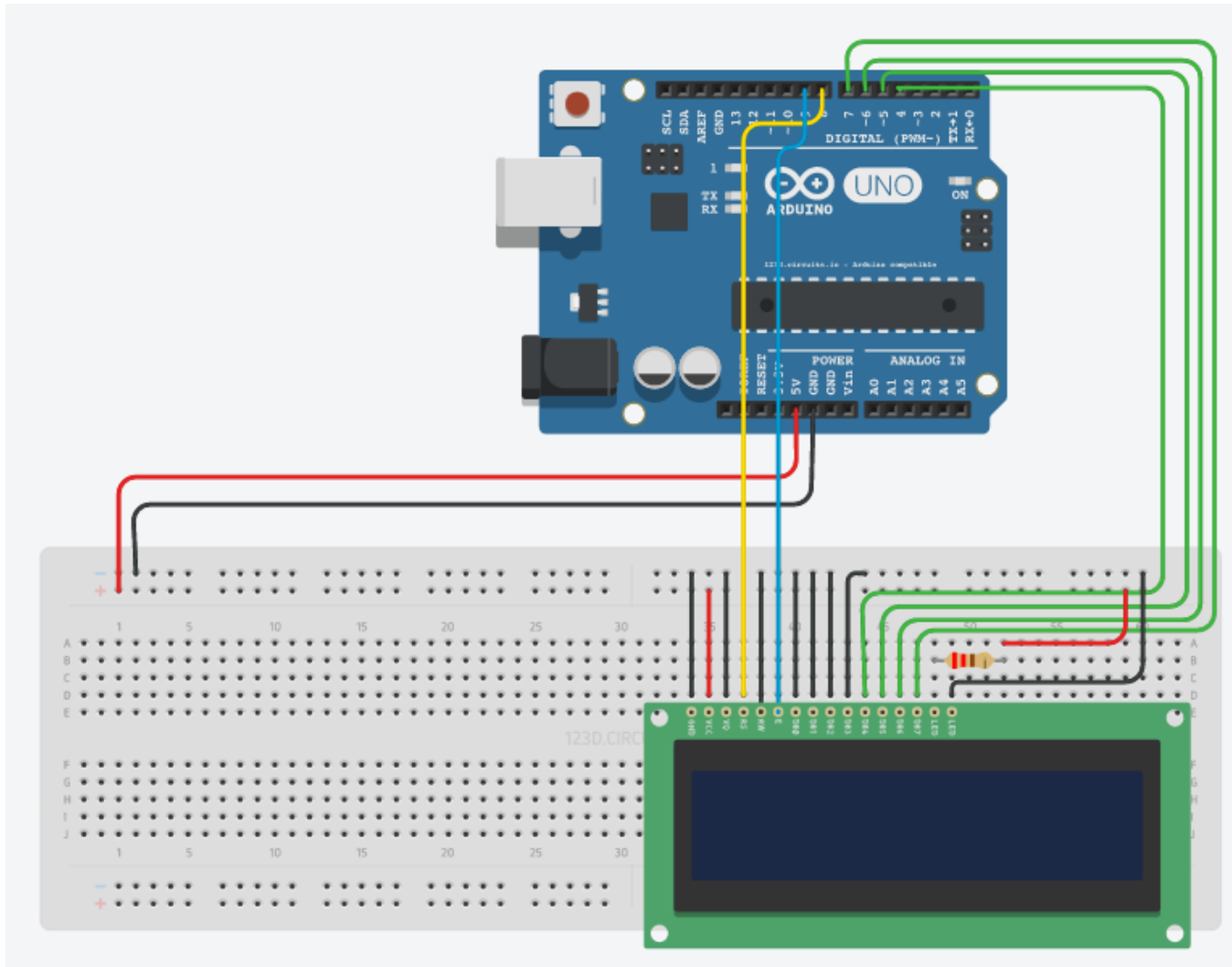
NIBBLE BAIXO \ NIBBLE ALTO		0_	1_	2_	3_	4_	5_	6_	7_	8_	9_	A_	B_	C_	D_	E_	F_
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
_0	xxxx 0000	CG RAM (1)			00PzP												
_1	xxxx 0001	CG RAM (2)			!1A0a9												
_2	xxxx 0010	CG RAM (3)			"2BRbr												
_3	xxxx 0011	CG RAM (4)			#3CScs												
_4	xxxx 0100	CG RAM (5)			\$4DTdt												
_5	xxxx 0101	CG RAM (6)			%5EUeu												
_6	xxxx 0110	CG RAM (7)			&6FUFv												
_7	xxxx 0111	CG RAM (8)			'7Gw9w												
_8	xxxx 1000	(1)			(8HXhx												
_9	xxxx 1001	(2))9IYiy												
_A	xxxx 1010	(3)			*:JZjz												
_B	xxxx 1011	(4)			+;Klk{												
_C	xxxx 1100	(5)			,<L*ll												
_D	xxxx 1101	(6)			-=MIm}												
_E	xxxx 1110	(7)			.>N^n+												
_F	xxxx 1111	(8)			/?O_o+												

LCD_4bits_ident_num.c

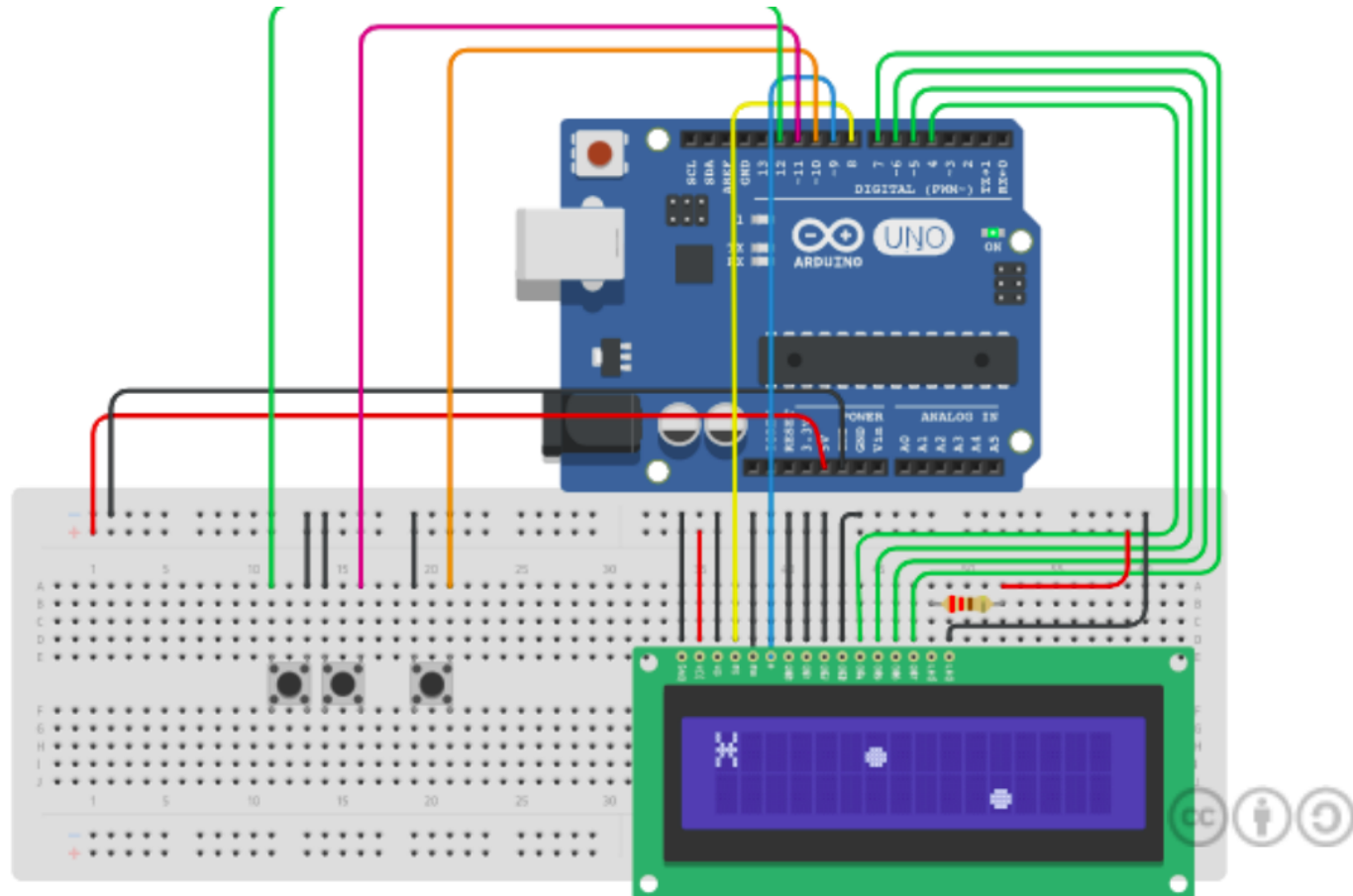
```
//-----  
// AVR e Arduino: Técnicas de Projeto, 2a ed. - 2012. //  
//-----  
//===== //  
// ACIONANDO UM DISPLAY DE CRISTAL LIQUIDO DE 16x2 //  
// Uso da função ident_num(...) //  
//===== //  
  
#include "def_principais.h" //inclusão do arquivo com as principais definições  
#include "LCD.h"  
  
//-----  
int main()  
{  
    unsigned char digitos[tam_vetor]; //declaração da variável para armazenagem dos digitos  
    unsigned char cont;  
  
    DDRD = 0xFF; //PORTD como saída  
    DDRB = 0xFF;  
  
    inic_LCD_4bits(); //inicializa o LCD  
  
    while(1)  
    {  
        for(cont=0; cont<101; cont++)  
        {  
            ident_num(cont,digitos);  
            cmd_LCD(0x8D,0); //desloca o cursor para que os 3 digitos fiquem a direita do LCI  
            cmd_LCD(digitos[2],1);  
            cmd_LCD(digitos[1],1);  
            cmd_LCD(digitos[0],1);  
            _delay_ms(200); //tempo para a troca de valor  
        }  
    }  
}
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Linha 1	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Linha 2	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

<https://www.tinkercad.com/things/e2pd20d7JH5>



<https://www.tinkercad.com/things/18HseVVcMzj>



Referências

- **AVR e Arduino – Técnicas de Projeto**
- **Simulador LCD**

<http://www.dinceraydin.com/djlcddsim/djlcddsim.html>