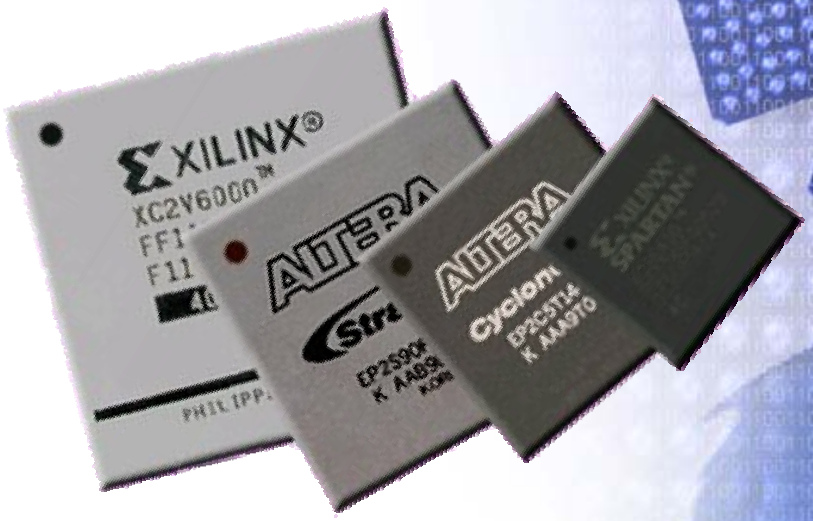
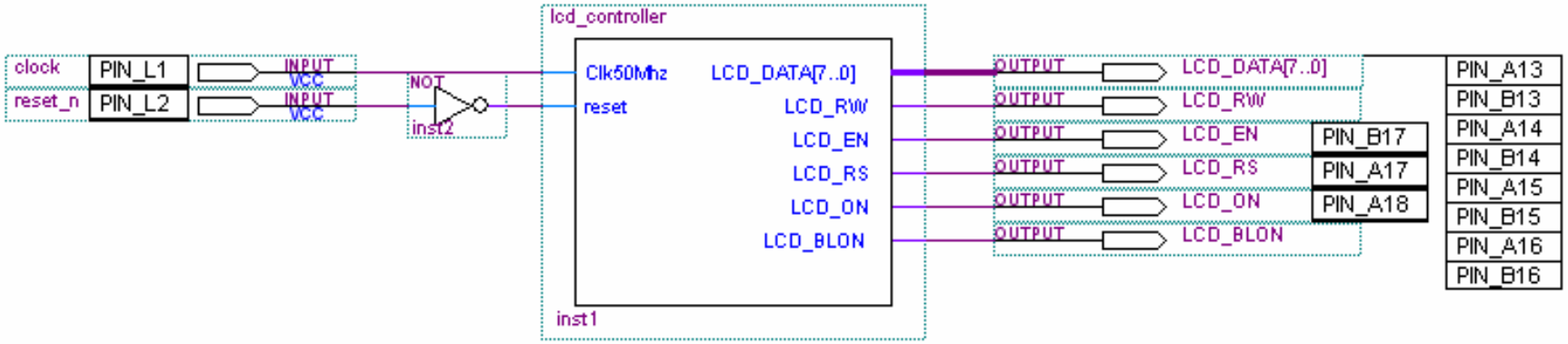
Realizzazione di un LCD controller x HD44780 Hitachi



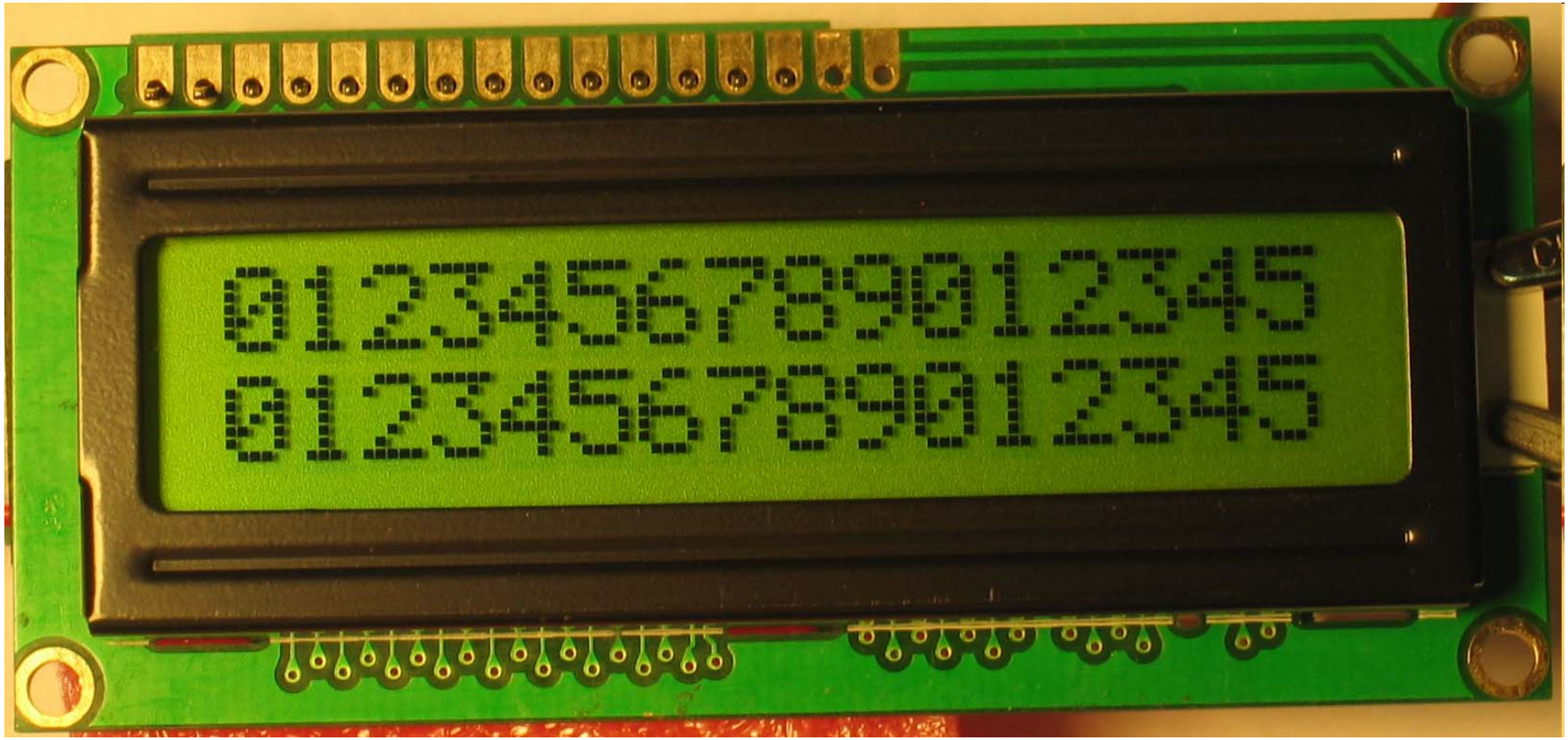
Giovanni De Luca

LCD HD44780 16x2



	Node Name	Direction	Location
1	clock	Input	PIN_L1
2	LCD_BLON	Output	
3	LCD_DATA[7]	Output	PIN_B16
4	LCD_DATA[6]	Output	PIN_A16
5	LCD_DATA[5]	Output	PIN_B15
6	LCD_DATA[4]	Output	PIN_A15
7	LCD_DATA[3]	Output	PIN_B14
8	LCD_DATA[2]	Output	PIN_A14
9	LCD_DATA[1]	Output	PIN_B13
10	LCD_DATA[0]	Output	PIN_A13
11	LCD_EN	Output	PIN_A17
12	LCD_ON	Output	
13	LCD_RS	Output	PIN_A18
14	LCD_RW	Output	PIN_B17
15	reset_n	Input	PIN_L2
16	<<new node>>		

LCD HD44780 16x2



Pin collegamenti LCD

Table 2.1., Pin assignment for <= 80 character displays

Pin number	Symbol	Level	I/O	Function
1	Vss	-	-	Power supply (GND)
2	Vcc	-	-	Power supply (+5V)
3	Vee	-	-	Contrast adjust
4	RS	0/1	I	0 = Instruction input 1 = Data input
5	R/W	0/1	I	0 = Write to LCD module 1 = Read from LCD module
6	E	1, 1->0	I	Enable signal
7	DB0	0/1	I/O	Data bus line 0 (LSB)
8	DB1	0/1	I/O	Data bus line 1
9	DB2	0/1	I/O	Data bus line 2
10	DB3	0/1	I/O	Data bus line 3
11	DB4	0/1	I/O	Data bus line 4
12	DB5	0/1	I/O	Data bus line 5
13	DB6	0/1	I/O	Data bus line 6
14	DB7	0/1	I/O	Data bus line 7 (MSB)

Instruction SET

2.2. Instruction set

Table 2.3. HD44780 instruction set

Instruction	Code										Description	Execution time**
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears display and returns cursor to the home position (address 0).	1.64mS
Cursor home	0	0	0	0	0	0	0	0	1	*	Returns cursor to home position (address 0). Also returns display being shifted to the original position. DDRAM contents remains unchanged.	1.64mS
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction (I/D), specifies to shift the display (S). These operations are performed during data read/write.	40uS
Display On/Off control	0	0	0	0	0	0	1	D	C	B	Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	40uS
Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	Sets cursor-move or display-shift (S/C), shift direction (R/L). DDRAM contents remains unchanged.	40uS
Function set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length (DL), number of display line (N) and character font(F).	40uS
Set CGRAM address	0	0	0	1	CGRAM address						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40uS
Set DDRAM address	0	0	1	DDRAM address							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40uS
Read busy-flag and address counter	0	1	BF	CGRAM / DDRAM address							Reads Busy-flag (BF) indicating internal operation is being performed and reads CGRAM or DDRAM address counter contents (depending on previous instruction).	0uS
Write to CGRAM or DDRAM	1	0	write data								Writes data to CGRAM or DDRAM.	40uS
Read from CGRAM or DDRAM	1	1	read data								Reads data from CGRAM or DDRAM.	40uS

Remarks:
- DDRAM = Display Data RAM.
- CGRAM = Character Generator RAM.
- DDRAM address corresponds to cursor position.
- * = Don't care.
- ** = Based on F_{osc} = 250kHz.

Bit names

Table 2.4. Bit names

Bit name	Setting / Status	
I/D	0 = Decrement cursor position	1 = Increment cursor position
S	0 = No display shift	1 = Display shift
D	0 = Display off	1 = Display on
C	0 = Cursor off	1 = Cursor on
B	0 = Cursor blink off	1 = Cursor blink on
S/C	0 = Move cursor	1 = Shift display
R/L	0 = Shift left	1 = Shift right
DL	0 = 4-bit interface	1 = 8-bit interface
N	0 = 1/8 or 1/11 Duty (1 line)	1 = 1/16 Duty (2 lines)
F	0 = 5x7 dots	1 = 5x10 dots
BF	0 = Can accept instruction	1 = Internal operation in progress

DDRAM 1-line

Table 2.5. DDRAM address usage for a 1-line LCD

Display size	Visible	
	Character positions	DDRAM addresses
1*8	00..07	0x00..0x07
1*16	00..15	0x00..0x0F [1] [2] [3] [4]
1*20	00..19	0x00..0x13
1*24	00..23	0x00..0x17
1*32	00..31	0x00..0x1F
1*40	00..39	0x00..0x27

DDRAM 2-line

Table 2.6. DDRAM address usage for a 2-line LCD

Display size	Visible	
	Character positions	DDRAM addresses
2*16	00..15	0x00..0x0F + 0x40..0x4F [1]
2*20	00..19	0x00..0x13 + 0x40..0x53
2*24	00..23	0x00..0x17 + 0x40..0x57
2*32	00..31	0x00..0x1F + 0x40..0x5F
2*40	00..39	0x00..0x27 + 0x40..0x67

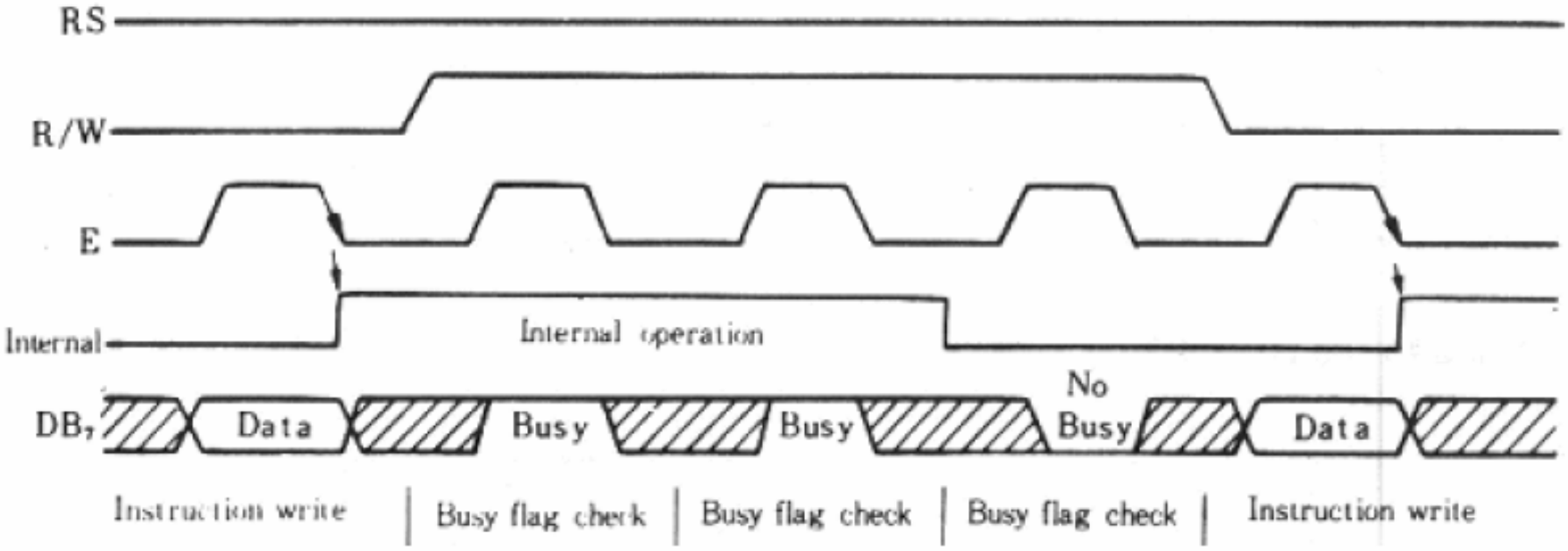
Interfacing 8-bit

2.4. Interfacing

TOC

2.4.1. 8-bit interface

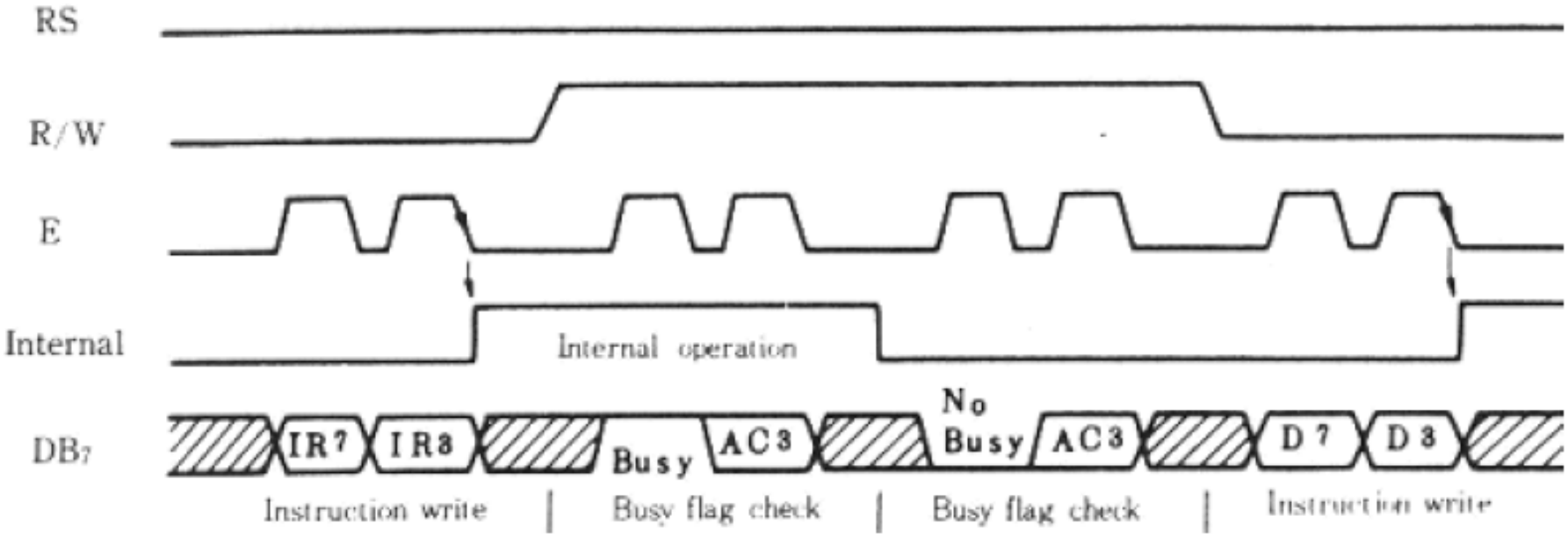
Example of busy flag testing using an 8-bit interface.



Interfacing 4-bit

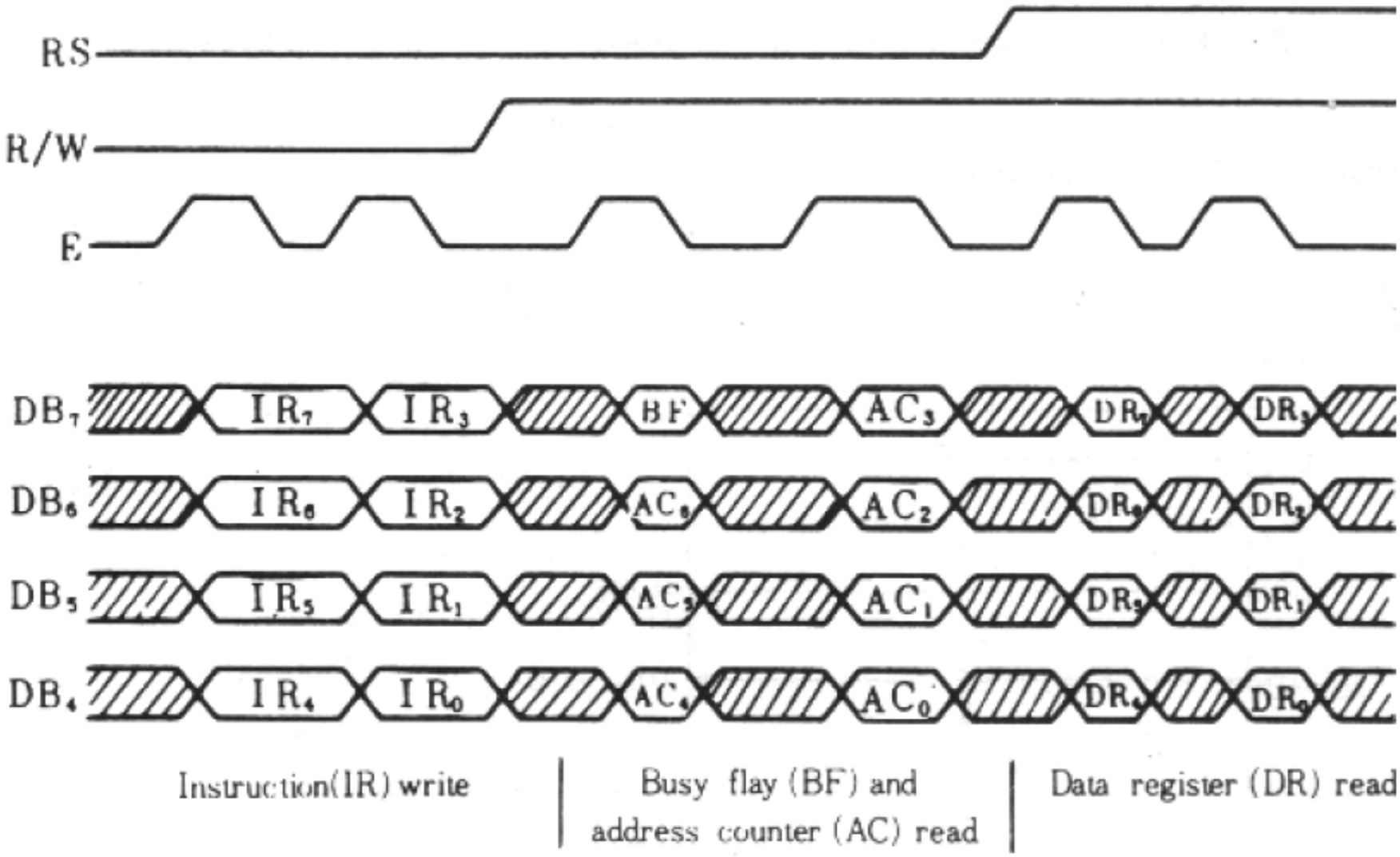
2.4.2. 4-bit interface

Example of busy flag testing using a 4-bit interface.



Transfer using 4-bit interface

Example of data transfer using a 4-bit interface.



ASCII Table (base)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

VHDL 1/6

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY lcd_controller IS
PORT (
   Clk50Mhz: IN STD_LOGIC;
   Reset: IN STD_LOGIC;
   LCD_DATA: OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
   LCD_RW, LCD_EN, LCD_RS: OUT STD_LOGIC;
   LCD_ON, LCD_BLON: OUT STD_LOGIC);
END lcd_controller;

ARCHITECTURE FSMD OF lcd_controller IS
   TYPE state_type IS (s1,s2,s3,s4,s10,s11,s12,s13,s20,s21,s22,s23,s24);
   SIGNAL state: state_type;

   CONSTANT max: INTEGER := 50000;
   CONSTANT half: INTEGER := max/2;
   SIGNAL clockticks: INTEGER RANGE 0 TO max;
   SIGNAL clock: STD_LOGIC;

   SUBTYPE ascii IS STD_LOGIC_VECTOR(7 DOWNTO 0);
   TYPE charArray IS array(1 to 16) OF ascii; -- questa è la matrice dei caratteri
   TYPE initArray IS array(1 to 7) OF ascii;
```

VHDL 2/6

```
    -- LCD initialization sequence codes
    -- 0x38 init four times
    -- 0x06 Entry mode set: Increment One; No Shift
    -- 0x0F Display control: Display ON; Cursor ON; Blink ON
    -- 0x01 Display clear
    CONSTANT initcode: initArray := (x"38",x"38",x"38",x"38",x"06",x"0F",x"01");

    -- "Giovanni"
    CONSTANT line1: charArray    := (x"20",x"20",x"20",x"67",x"69",x"6f",x"76",
                          x"61",x"6e",x"6e",x"69",x"20",x"20",x"20",x"20",x"20");

    -- "1234567890123456"
    CONSTANT line2: charArray    := (x"31",x"32",x"33",x"34",x"35",x"36",x"37",
                           x"38",x"39",x"30",x"31",x"32",x"33",x"34",x"35",x"36");

    SIGNAL count: INTEGER;

BEGIN

    LCD_ON <= '1';
    LCD_BLON <= '0';
```

VHDL 3/6

```
lcd_control: PROCESS(clock, reset)
   BEGIN
       IF(Reset = '1') THEN
           count <= 1;
           state <= s1;
       ELSIF(clock'EVENT AND clock = '1') THEN
           CASE state IS

           -- LCD initialization sequence
           -- The LCD_DATA is written to the LCD at the falling edge of the E line
           -- therefore we need to toggle the E line for each data write
           WHEN s1 =>
               LCD_DATA <= initcode(count);
               LCD_EN <= '1';   -- EN=1;
               LCD_RS <= '0';   -- RS=0; an instruction
               LCD_RW <= '0';   -- R/W'=0; write
               state <= s2;

           WHEN s2 =>
               LCD_EN <= '0';   -- set EN=0;
               count <= count + 1;
               IF count + 1 <= 7 THEN
                   state <= s1;
               ELSE
                   state <= s10;
               END IF;
```

VHDL 4/6

```
-- move cursor to first line of display
WHEN s10 =>
     LCD_DATA <= x"80";  -- x80 is address of 1st position on first line
     LCD_EN <= '1';   -- EN=1;
     LCD_RS <= '0';   -- RS=0; an instruction
     LCD_RW <= '0';   -- R/W'=0; write
     state <= s11;

WHEN s11 =>
     LCD_EN <= '0';   -- EN=0; toggle EN
     count <= 1;
     state <= s12;

-- write 1st line text
WHEN s12 =>
     LCD_DATA <= line1(count);
     LCD_EN <= '1';   -- EN=1;
     LCD_RS <= '1';   -- RS=1; data
     LCD_RW <= '0';   -- R/W'=0; write
     state <= s13;

WHEN s13 =>
     LCD_EN <= '0';   -- EN=0; toggle EN
     count <= count + 1;
     IF count + 1 <= 16 THEN
          state <= s12;
     ELSE
          state <= s20;
     END IF;
```

VHDL 5/6

```
-- move cursor to second line of display
WHEN s20 =>
     LCD_DATA <= x"BF";       -- xBF is address of 1st position on second line
     LCD_EN <= '1';   -- EN=1;
     LCD_RS <= '0';   -- RS=0; an instruction
     LCD_RW <= '0';   -- R/W'=0; write
     state <= s21;

WHEN s21 =>
     LCD_EN <= '0';   -- EN=0; toggle EN
     count <= 1;
     state <= s22;

-- write 2nd line text
WHEN s22 =>
     LCD_DATA <= line2(count);
     LCD_EN <= '1';   -- EN=1;
     LCD_RS <= '1';   -- RS=1; data
     LCD_RW <= '0';   -- R/W'=0; write
     state <= s23;

WHEN s23 =>
     LCD_EN <= '0';   -- set EN=0;
     count <= count + 1;
     IF count + 1 <= 16 THEN
          state <= s22;
     ELSE
          state <= s24;
     END IF;
```

VHDL 6/6

```
            WHEN s24 =>
                state <= s24;

            WHEN OTHERS =>
                state <= s24;
            END CASE;
        END IF;
    END PROCESS;
-------------------------------------------------------
ClockDivide: PROCESS
    BEGIN
        WAIT UNTIL Clk50Mhz'EVENT and Clk50Mhz = '1';

        IF clockticks < max THEN    -- if < 50000
            clockticks <= clockticks + 1;
        ELSE
            clockticks <= 0;
        END IF;

        IF clockticks < half THEN
            clock <= '0';
        ELSE
            clock <= '1';
        END IF;
    END PROCESS;
-------------------------------------------------------
END FSMD;
```