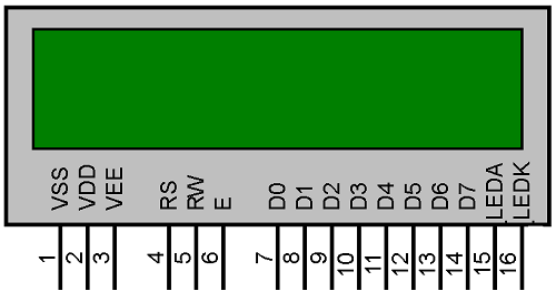
ACKNOWLEDGEMENT

I would like to express my gratitude toward our group members: Angelo, Federico and Walid

Greate appreciation to our kind supervisors: Zubat and Ludovico

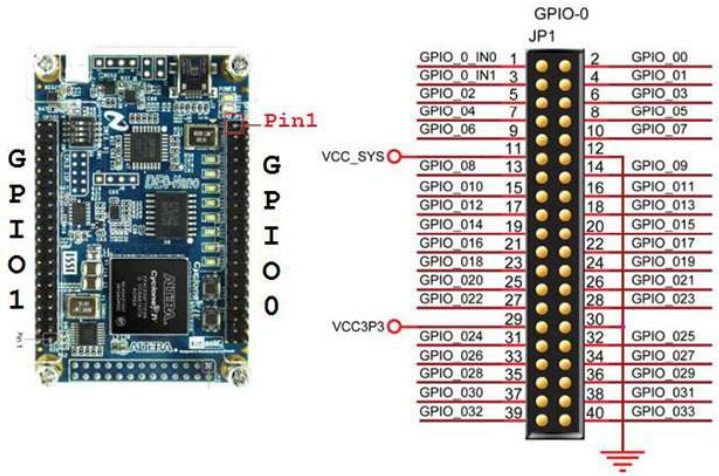
lcd1602 and how it works:

Schematic:

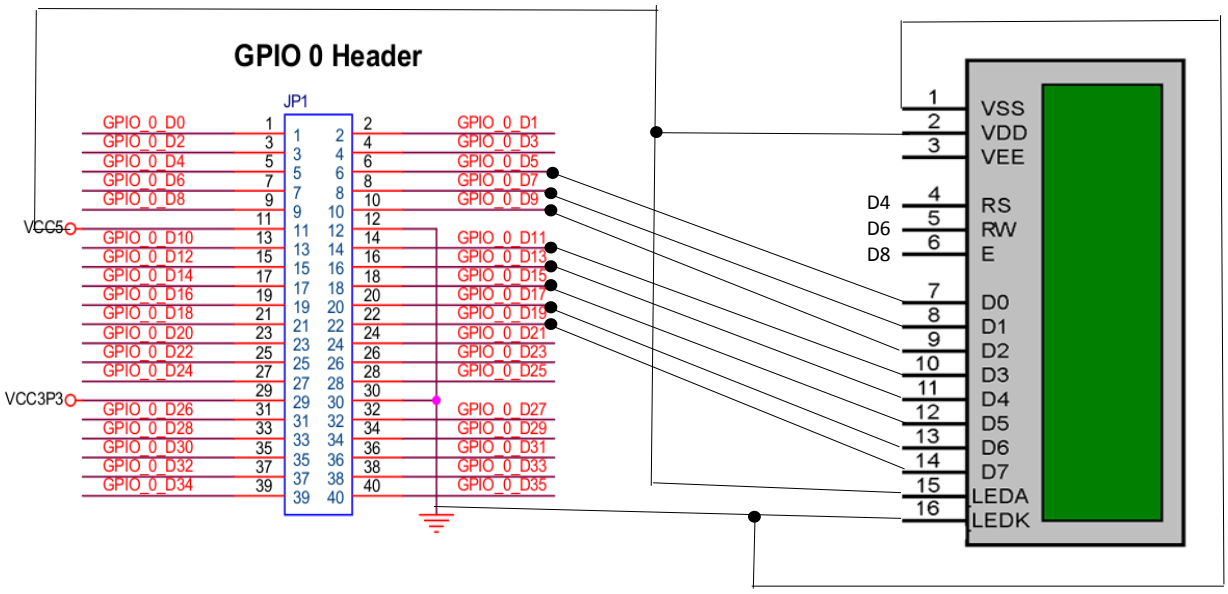


Pin definitions:
Pin 1 (Vss): Function as Ground Terminal.
Pin 2 (Vdd): Function as Positive Supply (2.7V to 5.5V).
Pin 3 (Vee): Function as Contrast adjustment (Ground to Vcc) .
Pin 4 (RS): Function as Register Select (If 0 is refer to Instruction Register and if 1 is refer toData Register).
Pin 5 (R/W): Its function to Read or Write Signal (if 1 mean to Read and if 0
mean to Write).
Pin 6 (E): Function as Enable (1) or Falling edge (0).
Pin 7 to Pin 14 (DB0 – DB7) : Refer to Bi-directional data bus, data transfer is performed one, through DB0 to DB7, in this case of interface data length is 8-bits; and twice, through DB4 to DB7 in this case of interface data length is 4-bits (Upper four bits first and then Lower four bits more).
Pin 15 (K): Function to Back light LED cathode terminal.
Pin 16 (A): Function to Back light LED anode terminal.

How we connect it to DE0-NANO:
We connected pins according to what is mentioned in the DE0-NANO manual, here is the picture of the port which we used:



Here is the connection map between LCD and GPIO-0:



Important observation: please take in to account that ground for both FPGA and LCD should be the same otherwise LCD won't be initialized.

How to drive LCD:

In order to be able to send characters to be shown on LCD firstly, we need to send some initialization instruction to LCD. To do so we should check the datasheet of the LCD first, then start sending instruction according to following steps for each instruction:

1- Send bit stream of 8 bits instruction on data pins and RS=0, R/W=0, E=0
2- Keep the same stream of 8 bits of instruction on data pins and, RS=0, R/W=0, E=1
3- It is mandatory to wait for at least more than 40 us (processing time of each instruction)

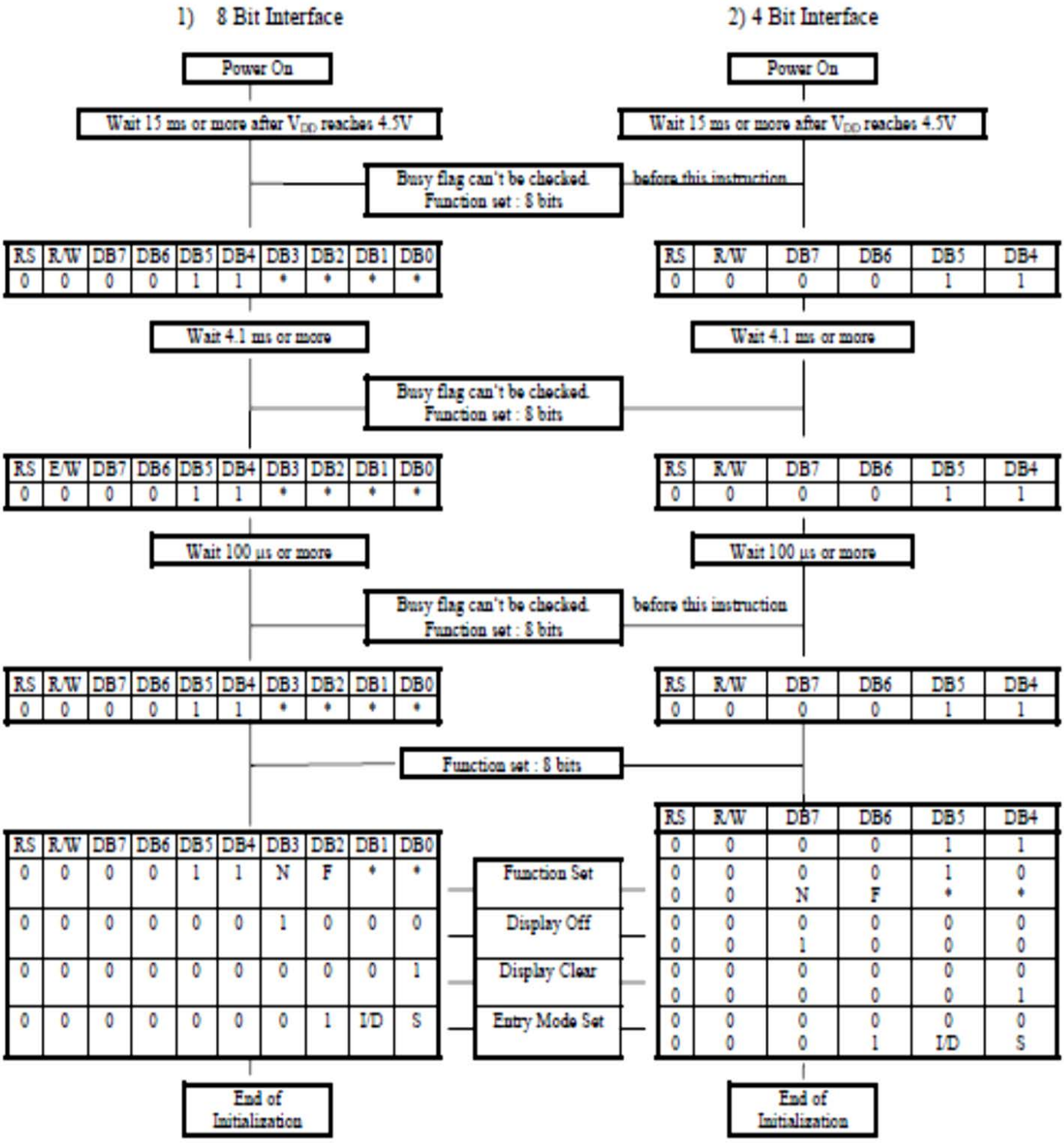
Initialization commands:

In page 4 a list of initialization commands is tabulated and in page 5 a flowchart proposing how to do initialization steps is depicted and in page 6 there is a tabulated example and from the page 7 on there is a sample VHDL code showing how to do initialization and write some characters on LCD with a Finite State Machine.

Instruction Set

FUNCTION	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION	EXECU. TIME* (MAX.)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and returns the cursor to home position (address 0).	1.64ms
Return Home	0	0	0	0	0	0	0	0	1	x	Return the cursor to the home position. Also returns the display being shifted to the original position. DD RAM contents remain unchanged.	1.64ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Set cursor move direct and specifies display shift.These operations are performed during data rite/read. For normal operation, set S to zero. I/D=1 : increment ; 0 :decrement ;S=1 : accompanies display shift when data is written, for normal operation, set to zero.	40 μs
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF all display(D),cursor ON/OFF(C), and blink of cursor position character(B). D=1: ON display; 0:OFF display. C=1: ON cursor;0: OFF cursor. B=1: ON blink cursor; 0: OFF blink cursor.	40 μs
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	x	x	Move the cursor and shift the display without changing DD RAM contents. S/C=1: Display shift; 0:Cursor move. R/L=1: shift to right; 0: shift to left.	40 μs
Function Set	0	0	0	0	1	DL	N	F	x	x	Set the interface data length (DL). Number of display lines (N) and character font (F). DL=1: 8 bits; 0:4 bits. N=1: 2 lines; 0: 1 lines. F=1: 5x10 dots; 0: 5x7 dots.	40 μs
Set CG RAM address	0	0	0	1	ACG						Set CG RAM address. CG RAM data is sent and received after this setting.	40 μs
Set DD RAM address	0	0	1	ADD							Set DD RAM address. DD RAM data is sent and received after this setting	40 μs
Read busy flag & address	0	1	BF	AC							Reads Busy Flag (BF) indicating internal operation is being performed and reads address counter contents. BF=1: internally operating. 0: can accept instruction	1 μs
Write Data to CG/DDRAM	1	0	WRITE DATA								Write data into DD RAM or CG RAM.	40 μs
Read Data for CG/DDRAM	1	1	READ DATA								Read data from DD RAM or CG RAM	40 μs

Bit		Settings
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


1) 8 Bit Interface
2) 4 Bit Interface
Power On
Wait 15 ms or more after V_{DD} reaches 4.5V
Busy flag can't be checked.
Function set : 8 bits
before this instruction
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
Wait 4.1 ms or more
Busy flag can't be checked.
Function set : 8 bits
RS E/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
Wait 100 μs or more
Busy flag can't be checked.
Function set : 8 bits
before this instruction
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 * * * *
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
Function set : 8 bits
RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
0 0 0 0 1 1 N F * *
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 I/D S
Function Set
Display Off
Display Clear
Entry Mode Set
RS R/W DB7 DB6 DB5 DB4
0 0 0 0 1 1
0 0 0 0 1 0
0 0 N F * *
0 0 0 0 0 0
0 0 1 0 0 0
0 0 0 0 0 1
0 0 0 0 0 0
0 0 0 1 I/D S
End of Initialization
End of Initialization

Software Example

8-bit operation (8 bits 2 lines)

Function	RS	Rw	D7	D6	D5	D4	D3	D2	D1	D0	Display	Description
Power on delay												Initialization. No display appears.
Function set	0	0	0	0	1	1	0	0	x	x		Sets to 8-bit operation and selects 2-line display and 5x7 dots character font. (Note: number of display lines and character fonts cannot be chang after this.)
Display OFF	0	0	0	0	0	0	1	0	0	0		Turn off display.
Display ON	0	0	0	0	0	0	1	1	1	0	_	Turn on display and cursor
Entry Mode Set	0	0	0	0	0	0	0	1	1	0	_	Set mode to increment the address by one and to shift the cursor to the right, at the time of write, to the DD/CG RAM Display is not shifted.
Write data to CG/DD RAM	1	0	0	1	0	1	0	0	1	1	S_	Write "S". Cursor incremented by one and shift to right.
Write data to CG/DD RAM	1 1 1	0 0 0	0 0 0	1 1 1	0 0 0	0 0 0	0 0 0	1 1 0	0 0 1	0 1 1	SDEC_	Write "D" , "E" , and "C".
Set DD RAM	0	0	1	1	0	0	0	0	0	0	SDEC _	Set RAM address so that the cursor is propositioned at the head of the second line.
Write data to CG/DD RAM					* *						SDEC CR_	Write "C" , and "R".
Cursor or display shift	0	0	0	0	0	1	0	0	x	x	SDEC CR	Shift only the cursor position to the left.
Write data to CG/DD RAM					* *						SDEC CO., LTD._	Write "O., LTD." .
Entry Mode Set	0	0	0	0	0	0	0	1	1	1	SDEC CO., LTD._	Set display mode shift at the time during writing operation.
Write data to CG/DD RAM	1	0	0	1	1	1	1	0	0	0	DEC O., LTD. x_	Write " x". Cursor incremented by one and shift to right. (The display move to left.)
Write data to CG/DD RAM					* *							Write other characters.
Return Home	0	0	0	0	0	0	0	0	1	0	SDEC CO., LTD.	Return both display and cursor to the original position (Set address to zero).

VHDL sample code:

```
	--Necessary Header Files
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

	--Define The Core Entity
ENTITY LCD IS
PORT(
		--Counter/VGA Timing
		CLK		: IN STD_LOGIC;

		--LCD Control Signals
		LCD_ENABLE	: OUT STD_LOGIC;
		LCD_RW	: OUT STD_LOGIC;
		LCD_RS	: OUT STD_LOGIC;

		--LCD Data Signals
		LCD_DATA	: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
end LCD;

	--Define The Architecture Of The Entity
ARCHITECTURE behavior of LCD IS

type state_type is (	S0, S1, S2, S3, S4, S5, S6, S7, S8, S9,
					S10, S11, S12, S13, S14, S15, S16, S17, S18, S19,
					S20, S21, S22, S23, S24, S25, S26, S27, S28, S29,
					S30, S31, S32, S33, S34, S35, S36, S37, S38, S39,
					S40,S41,S42,S43,S44,S45,S46,S47,S48,IDLE);
signal next_state: state_type;

BEGIN

PROCESS
VARIABLE cnt: INTEGER RANGE 0 TO 17500000;
BEGIN
```

```
WAIT UNTIL(clk'EVENT) AND (clk = '1');

	--Count Clock Ticks
	IF(cnt = 1750000)THEN
		cnt := 0;
	ELSE
		cnt := cnt  + 1;
	END IF;

	--Slowly Move Into Next States
	IF(cnt = 1500000)THEN
	--Next State Logic

		case next_state is


------------------Function Set------------------
			when S0 =>
				next_state <= S1;

				LCD_DATA	<= "00111000";

				LCD_ENABLE	<= '0';
				LCD_RW		<= '0';
				LCD_RS		<= '0';

			when S1 =>
				next_state <= S2;

				LCD_DATA	<= "00111000";

				LCD_ENABLE	<= '1';
				LCD_RW		<= '0';
				LCD_RS		<= '0';

			when S2 =>
				next_state <= S3;

				LCD_DATA	<= "00111000";
```

```
                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

------------------Reset Display------------------
                    when S3 =>
                        next_state <= S4;

                        LCD_DATA        <= "00000001";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

                    when S4 =>
                        next_state <= S5;

                        LCD_DATA        <= "00000001";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

                    when S5 =>
                        next_state <= S6;

                        LCD_DATA        <= "00000001";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';


------------------Display On------------------
                    when S6 =>
                        next_state <= S7;

                        LCD_DATA        <= "00001110";
```

```
                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '0';

            when S7 =>
                next_state <= S8;

                LCD_DATA        <= "00001110";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '0';

            when S8 =>
                next_state <= S9;

                LCD_DATA        <= "00001110";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

----------Write 'R'----------
            when S9 =>
                next_state <= S10;

                LCD_DATA        <= x"52";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S10 =>
                next_state <= S11;

                LCD_DATA        <= x"52";

                LCD_ENABLE      <= '0';
```

```
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'A'------------------
            when S11 =>
                next_state <= S12;

                LCD_DATA        <= X"41";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S12 =>
                next_state <= S13;

                LCD_DATA        <= X"41";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'M'------------------
            when S13 =>
                next_state <= S14;
                LCD_DATA        <= X"4D";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S14 =>
                next_state <= S15;
                LCD_DATA        <= X"4D";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';
```

```
------------------WRITE 'blank'------------------
                when S15 =>
                        next_state <= S16;
                        LCD_DATA        <= X"10";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

                when S16 =>
                        next_state <= S17;
                        LCD_DATA        <= X"10";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

------------------WRITE 'T'------------------
                when S17 =>
                        next_state <= S18;
                        LCD_DATA        <= X"54";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

                when S18 =>
                        next_state <= S19;
                        LCD_DATA        <= X"54";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

------------------WRITE 'E'------------------
                when S19 =>
                        next_state <= S20;
                        LCD_DATA        <= X"45";
```

```
                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S20 =>
                next_state <= S21;
                LCD_DATA        <= X"45";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'S'------------------
            when S21 =>
                next_state <= S22;
                LCD_DATA        <= X"53";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S22 =>
                next_state <= S23;
                LCD_DATA        <= X"53";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'T'------------------
            when S23 =>
                next_state <= S24;
                LCD_DATA        <= X"54";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S24 =>
```

```
                next_state <= S25;
                LCD_DATA        <= X"54";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'E'------------------
            when S25 =>
                next_state <= S26;
                LCD_DATA        <= X"45";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S26 =>
                next_state <= S27;
                LCD_DATA        <= X"45";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

------------------WRITE 'R'------------------
            when S27 =>
                next_state <= S28;
                LCD_DATA        <= X"52";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S28 =>
                next_state <= S29;
                LCD_DATA        <= X"52";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
```

```
                        LCD_RS          <= '1';

------------------WRITE 'new line'-------------------
                    when S29 =>
                        next_state <= S30;

                        LCD_DATA        <= "11000000";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

                    when S30 =>
                        next_state <= S31;

                        LCD_DATA        <= "11000000";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

                    when S31 =>
                        next_state <= S32;

                        LCD_DATA        <= "11000000";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '0';

------------------WRITE 'W'-------------------
                    when S32 =>
                        next_state <= S33;
                        LCD_DATA        <= X"57";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';
```

```
                        when S33 =>
                                next_state <= S34;
                                LCD_DATA        <= X"57";

                                LCD_ENABLE      <= '0';
                                LCD_RW          <= '0';
                                LCD_RS          <= '1';

------------------WRITE 'E'------------------
                        when S34 =>
                                next_state <= S35;
                                LCD_DATA        <= X"45";

                                LCD_ENABLE      <= '1';
                                LCD_RW          <= '0';
                                LCD_RS          <= '1';

                        when S35 =>
                                next_state <= S36;
                                LCD_DATA        <= X"45";

                                LCD_ENABLE      <= '0';
                                LCD_RW          <= '0';
                                LCD_RS          <= '1';

------------------WRITE 'E'------------------2
                        when S36 =>
                                next_state <= S37;
                                LCD_DATA        <= X"45";

                                LCD_ENABLE      <= '1';
                                LCD_RW          <= '0';
                                LCD_RS          <= '1';

                        when S37 =>
                                next_state <= S38;
                                LCD_DATA        <= X"45";

                                LCD_ENABLE      <= '0';
```

```
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

------------------WRITE 'E'-------------------3
                when S38 =>
                        next_state <= S39;
                        LCD_DATA        <= X"45";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

                when S39 =>
                        next_state <= S40;
                        LCD_DATA        <= X"45";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';
------------------WRITE 'O'-------------------
                when S40 =>
                        next_state <= S41;
                        LCD_DATA        <= X"4F";

                        LCD_ENABLE      <= '1';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

                when S41 =>
                        next_state <= S42;
                        LCD_DATA        <= X"4F";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';

------------------WRITE 'P'-------------------
                when S42 =>
                        next_state <= S43;
```

```
                LCD_DATA        <= X"50";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S43 =>
                next_state <= S44;
                LCD_DATA        <= X"50";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';
        ----------------WRITE 'E'-------------------
            when S44 =>
                next_state <= S45;
                LCD_DATA        <= X"45";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S45 =>
                next_state <= S46;
                LCD_DATA        <= X"45";

                LCD_ENABLE      <= '0';
                LCD_RW          <= '0';
                LCD_RS          <= '1';
----------------WRITE 'N'-------------------
            when S46=>
                next_state <= S47;
                LCD_DATA        <= X"4E";

                LCD_ENABLE      <= '1';
                LCD_RW          <= '0';
                LCD_RS          <= '1';

            when S47 =>
```

```
                        next_state <= S48;
                        LCD_DATA        <= X"4E";

                        LCD_ENABLE      <= '0';
                        LCD_RW          <= '0';
                        LCD_RS          <= '1';
-------------------------------------------------

                    when S48 =>
                        next_state <= IDLE;

                    when IDLE   =>
                        next_state <= IDLE;

                when others =>
                        next_state <= IDLE;

            end case;

        END IF;

END PROCESS;

END behavior;
```