VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI-590018

NEW HORIZON COLLEGE OF ENGINEERING

AUTONOMOUS COLLEGE Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade

Mini Project Report

On

"Morse code converter "

Submitted in partial fulfillment for of the award of degree

Bachelor of Engineering

In

Electronics and Communication Engineering

BY

Aishwarya M-1NH18EC002

Aishwarya N-1NH18EC003

B Gowthami -1NH18EC020

Chandana C-1NH18EC023

Under the Guidance of
Dr. Sanjeev Sharma
HOD
Department of ECE
New Horizon college of Engineering

NEW HORIZON
COLLEGE OF ENGINEERING
AUTONOMOUS COLLEGE Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade

CERTIFICATE

Certified that the Mini-project entitled "Morse Code Convertor" is carried out by **Aishwarya M** bearing **USN:1NH18EC002**, **Aishwarya N** bearing **USN: 1NH18EC003, B Gowthami bearing USN:1NH18EC020** and **Chandana C** bearing **USN: 1NH18EC023** bonafide students of **NEW HORIZON COLLEGE OF ENGINEERING, BENGALURU**, in partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication of the Visvesvaraya Technological University,Belgavi during the year **2018-2019**.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library .The Mini-project report has been approved as it satisfies the academic requirements in respect of the Mini-project work prescribed for the said degree.

Signature of the HoD
Dr. Sanjeev Sharma
Prof&HoD
Dept. of ECE
NHCE, Bengaluru

Signature of the guide
Dr Sanjeev Sharma
HOD
Dept. of ECE
NHCE, Bengaluru

EXTERNAL VIVA

Name of the Examiners: Signature with date:

1.

2.

3.

Table of Contents

ACKNOWLEDGEMENT:

The satisfaction that accompany the successful completion of any task would be, but impossible without the mention of the people who made it possible, whose constant guidance and encouragement helped us succeed.

We thank **Dr. Mohan Manghnani**, Chairman of **New Horizon Educational Institution**, for providing necessary infrastructure and creating good environment.

We also record here the constant encouragement and facilities extended to us by **Dr.Manjunatha**, Principal, NHCE and **Dr. Sanjeev Sharma**, head of the department of Electronics and Communication Engineering. We extend sincere gratitude to them.

We sincerely acknowledge the encouragement, timely help and guidance to us by our beloved guid**e Dr sanjeev Sharma** to complete the project within stipulated time successfully.

Finally, a note of thanks to the teaching and non-teaching staff of electronics and communication department for their co-operation extended to us, who helped us directly or indirectly in this successful completion of mini project.

Aishwarya M-1NH18EC002

Aishwarya N-1NH18EC003

B Gowthami -1NH18EC020

Chandana C-1NH18EC023

Abstract

Our project implements a system that translates the TEXT to MORSE CODE. The input is alphabets the corresponding equivalent morse codes. The output can be observed as square wave with long and short on periods for dots ans dashes respectively. This output when connected to speaker gives audio morse code output.

Chapter:1
Introduction

Our project implements a system that converts the text to morse code . Telegraphs using morse code where the initial modes of electrical signal communications. The system converts the text input to equivalent morse codes. The output is taken in the form of sound signals . The purpose of this project is to design a system which accepts input from the keyboard and gives the output in the form of morse code to the speaker.

What is a morse code encoder?

Morse code is a encoding scheme used in communication that encodes the text characters to standard orderd sequence of dots and dashes as specified by the international council.
Morse code is named after Samuel.F.B.Morse who was the founder of telegraph.

The internalional morse code encodes all the 26 english alphabets along with a few arabic numerals along with a few sets of punctuations and procedural signals. It does not diffrentiate between upper case and lower case letters.
Morse code is usually transmitted by means of on-off keys or switches of the information carrying medium that is electric pulse, radio waves, visible light , sound waves.
To increase the efficiency of encoding , morse code was designed so that the length of each symbol is approximately equal to the inverse of frequency of its occurrence in text of the english language it represents.[1]

Standard specifications:

- Length of dot is one unit.
- Length of dash is three units.
- Space between parts of the same letter is one unit.
- Space between letters is three units.
- Space between words is seven units.

Fig:1.1: standard morse code table.

Chapter:2

Literature survey

Morse was invented by an american scientist called Samuel Finely Breese Morse. He was also a famous painter. Telegraph was sent by tapping out the code for each letter. In the olden days telegraph was the only means of long distance communication using electric pulse.
The original morse code as invented actually was quite different from todays standard followed.

Samuel Finley Breese Morse, OIC (April 27, 1791 – April 2, 1872) was an American painter and inventor. After having established his reputation as a portrait painter, in his middle age Morse contributed to the invention of a single-wire telegraph system based on European telegraphs. He was a co-developer of Morse code and helped to develop the commercial use of telegraphy.

As noted, in 1825 New York City had commissioned Morse to paint a portrait of Lafayette in Washington, DC. While Morse was painting, a horse messenger delivered a letter from his father that read, "Your dear wife is convalescent". The next day he received a letter from his father detailing his wife's sudden death. Morse immediately left Washington for his home at New Haven, leaving the portrait of Lafayette unfinished. By the time he arrived, his wife had already beenburied. Heartbroken that for days he was unaware of his wife's failing health and her death, he decided to explore a means of rapid long distance communication.

Morse received a patent for the telegraph in 1847, at the old Beylerbeyi Palace (the present Beylerbeyi Palace was built in 1861–1865 on the same location) in Istanbul, which was issued by Sultan Abdülmecid, who personally tested the new invention.He was elected an Associate Fellow of the American Academy of Arts and Sciences in 1849. The original patent went to the Breese side of the family after the death of Samuel Morse.

In 1856, Morse went to Copenhagen and visited the Thorvaldsens Museum, where the sculptor's grave is in the inner courtyard. He was received by King Frederick VII, who decorated him with the Order of the Dannebrog for the telegraph. Morse expressed his wish to donate his Thorvaldsen portrait from 1831 in Rome to the king.The Thorvaldsen portrait today belongs to Margrethe II of Denmark.

Fig:2.1: **Samuel Finley Breese Morse.**

Fig:2.2: The first telegraph machine.

Chapter :3

Technology used

Block diagram:

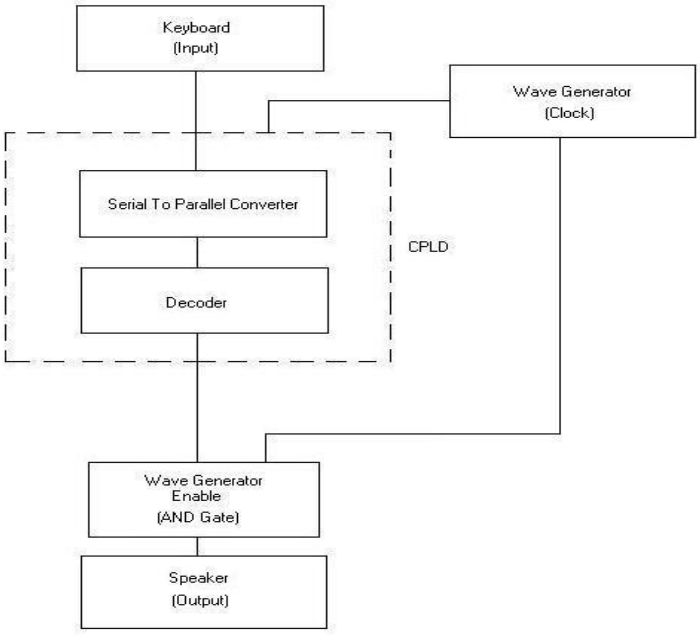

Fig:3.1 block diagram of the morse code encoder system

Discription of the block diagram

The input is given to the system using keyboard. The input is in the form of hexadecimal or binary equivalent values of the alphabets.
The second block is to convert the serial input of the data to parallel data to be readable by the program.
The decoder block converts the textto morse code with the help pf mux and other digital elements.
The corresponding wave form is generated and the output is given to the speaker.

“Decoder.vhd -- This is the main code that makes all the components work together, etc...

Stop.vhd -- This is the code for converting the serial data in from the keyboard to parallel data (a 7 bit binary number that represents a letter).

Pdatahold.vhd -- This code holds the parallel data.

Letters.vhd -- This code takes the 7 bit binary number that represents a letter and outputs the correct Morse code for the letter inputted”.

Soundout.vhd -- This code is used by the Letters.vhd. It is for outputting either a short pulse or a long pulse

Chapter :4

Proposed methodology

Circuit discription:

When the required letter is given as input from the PS/2 keyboard into the pogram. It converts the serial input data into parallel data.Each letter consists of seven binary digits , even the hexadecimal inpit is automatically converted into binary equvalent form.The output consists of series of morese code data.the output is fed to a speaker which produces short and long beeps for the corresponding letter input.
“A letter is inputted from a PS/2 keyboard into the CPLD. The CPLD converts the input from serial data into parallel data, each letter has a 7 bit binary number. The CPLD then decodes the 7 bit binary number and outputs the corresponding Morse code output for the particular letter as a series of 1's and 0's. The output of the CPLD is run through a 200 ohm resistor to reduce noise, and is ANDed with the output of a wave generator, the output of the AND gate is fed into a speaker which outputs a series of short and long beeps for the letter that was inputted”.

The code consists of five modules

Decoder- it is the main module which makes all the connections and interlinks the submodule to provide the desired output.

Stop-this module is used to convert the serial input of the data to parallel data.

Patahold-this code holds the parallel data until the exhicution of the system takes place , that is still the system identifies the data and maps it with the corresponding output.

Letters-this is the module is in which the actual morse code encoding takes place. It is the sole of the entire code. In this module the seven bit binary data is taken and the corresponding morse code is identified and the result is given from this module.

Soundout- this module is used to provide the output sound which is in the form of beeps. It taked the encoded morse code as input and produces the sound waves . it stimulates short pulse for dot and long pulse for dash.

PS/2 keyboard:
It is a 6 pin mini din connector keyborad .

Fig:4.1: the keyboard used to provide input to the system.

Letter	Hex (AT Keyboard)	Binary (AT Keyboard)	Morse Code Output
a	1C	0011100	.-
b	32	0110010	-...
c	21	0100001	-.-.
d	23	0100011	-..
e	24	0100100	.
f	2B	0101011	..-.
g	34	0110100	--.
h	33	0110011	....
i	43	1000011	..
j	3B	0111011	.---
k	42	1000010	-.-
l	4B	1001011	.-..
m	3A	0111010	--
n	31	0110001	-.
o	44	1000100	---
p	4D	1001101	.--.
q	15	0010101	--.-
r	2D	0101101	.-.
s	1B	0011011	...
t	2C	0101100	-
u	3C	0111100	..-
v	2A	0101010	...-
w	1D	0011101	.--
x	22	0100010	-..-
y	35	0110101	-.--
z	1A	0011010	--..
backspace	66	1100110	..... (to indicate a mistake)
space	29	0101001	(7 blank units)

Fig:4.2: binary and hexadecimal equivalent values of alphabets.

Chapter :5
Result obtained

The result of the project is matched with our expectations . the code is able to take the hexadecimal or binary input of data and give the corresponding morse code output . Due to limitted time and other factors we were not able to enode numbers in this particular code. This is restricted only to the encoding of alphabets.

We were not able to implement the code into the fpga due to lack available resources and hence the speaker output could not be experienced physically.

We were successfully able to convert our ideas to model and obtain the desired output.

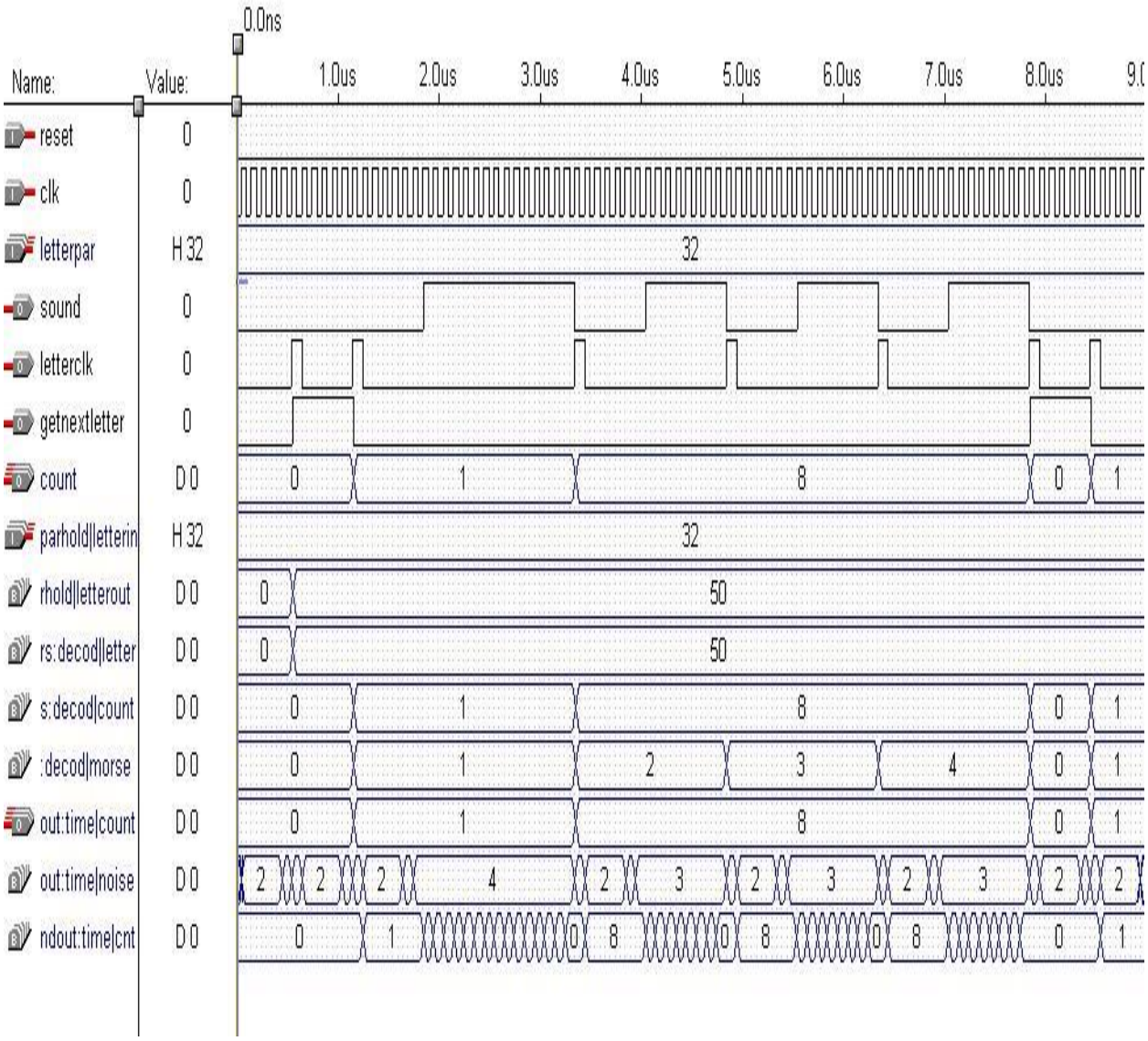

Fig:5.1: stimulation output for example 1

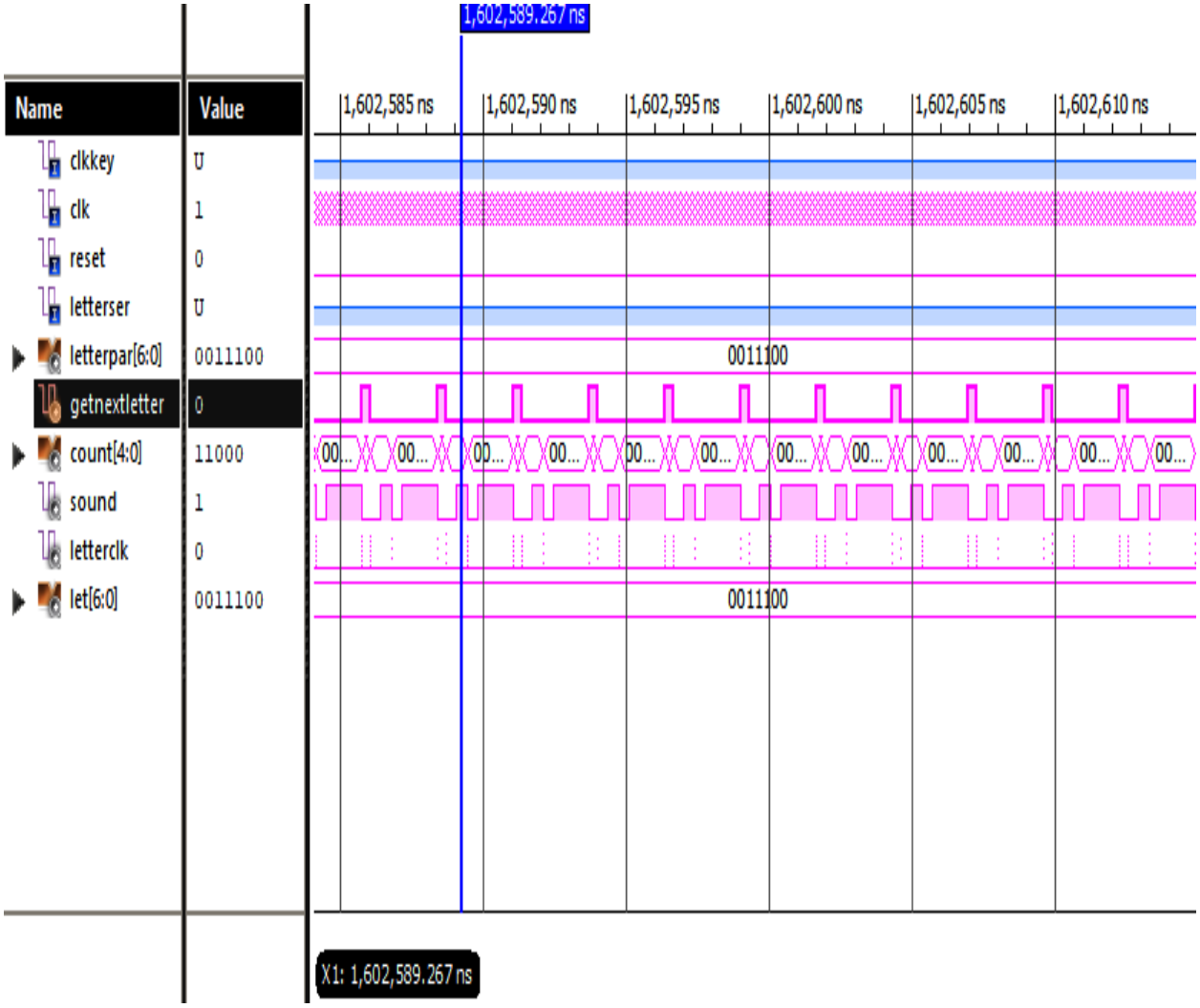

Fig:5.2:stimulation output for example 2

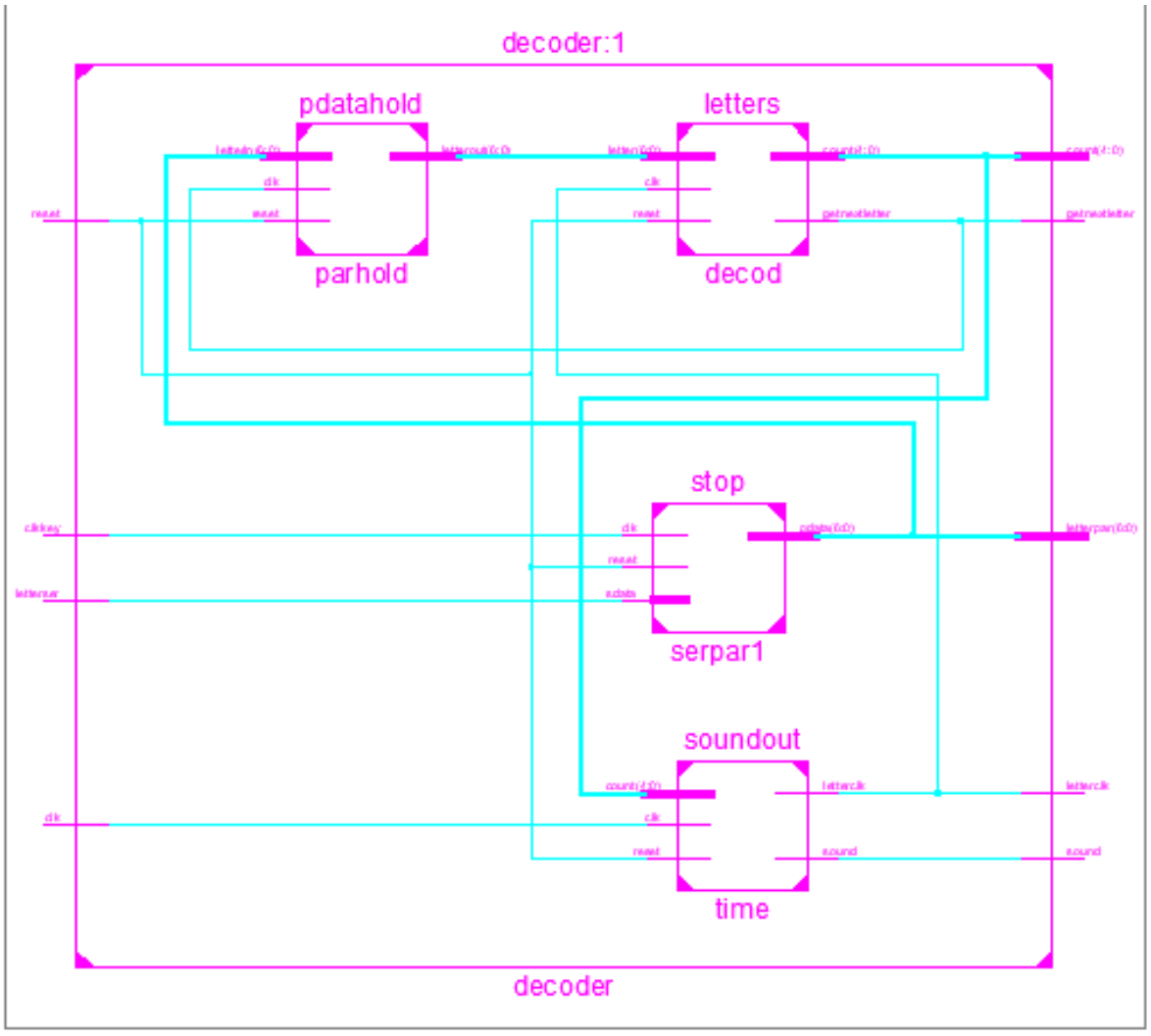

Fig:5.3:RTL schematic of the system.

Chapter : 6
Future scope

The morse codes are used in emergency situations during war for communication.
This is code language mainly used by the armed forces and the intelligence departnment.
In Aviation : during poor signal availability morse codes are send as electric pulses as it has less noise interferance.

During Natural calamities the rescue officers use morse code to communicate in unfavorable climate conditions using flash lights and other means which are available.
In aviation, pilots use radio navigation aids. To ensure that the stations the pilots are using are serviceable, the stations transmit a set of identification letters (usually a two-to-five-letter version of the station name) in Morse code. Station identification letters are shown on air navigation charts. For example, the VOR-DME based at Vilo Acuña Airport in Cayo Largo del Sur, Cuba is coded as "UCL", and UCL in Morse code is transmitted on its radio frequency. In some countries, during periods of maintenance, the facility may radiate a T-E-S-T code or the code may be removed which tells pilots and navigators that the station is unreliable. In Canada, the identification is removed entirely to signify the navigation aid is not to be used. In the aviation service, Morse is typically sent at a very slow speed of about 5 words per minute. In the U.S., pilots do not actually have to know Morse to identify the transmitter because the dot/dash sequence is written out next to the transmitter's symbol on aeronautical charts. Some modern navigation receivers automatically translate the code into displayed letters.

Morse code has been employed as an assistive technology, helping people with a variety of disabilities to communicate. For example, the Android operating system versions 5.0 and higher allow users to input text using Morse Code as an alternative to a keypad or handwriting recognition.

Morse can be sent by persons with severe motion disabilities, as long as they have some minimal motor control. An original solution to the problem that caretakers have to learn to decode has been an electronic typewriter with the codes written on the keys. Codes were sung by users; see the voice typewriter employing morse or votem, Newell and Nabarro, 1968.

Morse code can also be translated by computer and used in a speaking communication aid. In some cases, this means alternately blowing into and sucking on a plastic tube ("sip-and-puff" interface). An important advantage of Morse code over row column scanning is that once learned, it does not require looking at a display. Also, it appears faster than scanning."

In one case reported in the radio amateur magazine *QST*, an old shipboard radio operator who had a stroke and lost the ability to speak or write could communicate with his physician (a radio amateur) by blinking his eyes in Morse. Two examples of communication in intensive care units were also published in *QST*, Another example occurred in 1966 when prisoner of war Jeremiah

Denton, brought on television by his North Vietnamese captors, Morse-blinked the word *TORTURE*. In these two cases, interpreters were available to understand those series of eye-blinks.

Morse code can be transmitted in a number of ways: originally as electrical pulses along a telegraph wire, but also as an audio tone, a radio signal with short and long tones, or as a mechanical, audible, or visual signal (e.g. a flashing light) using devices like an Aldis lamp or a heliograph, a common flashlight, or even a car horn. Some mine rescues have used pulling on a rope - a short pull for a dot and a long pull for a dash.

Morse code is transmitted using just two states (on and off). Historians have called it the first digital code. Morse code may be represented as a binary code, and that is what telegraph operators do when transmitting messages. Working from the above ITU definition and further defining a bit as a dot time, a Morse code sequence may be made from a combination of the following five bit-strings.

The very long time constants of 19th and early 20th century submarine communications cables required a different form of Morse signalling. Instead of keying a voltage on and off for varying times, the dots and dahes were represented by two polarities of voltage impressed on the cable, for a uniform time.

References :

[1].”*international Morse code recommendation ITU*” october 2009.

[2]. *“Morse code generator using Microcontroller with alphanumeric keypad”* 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).

Apendix

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity decoder is
 port ( clkkey, clk, reset : in std_logic;
   letterser : in std_logic;
   letterpar : buffer std_logic_vector (6 downto 0);
   getnextletter : buffer std_logic;
   count : buffer std_logic_vector (4 downto 0);
   sound, letterclk : buffer std_logic );
end decoder;
architecture ltos of decoder is
 signal let : std_logic_vector (6 downto 0);
-- convert serial to parallel
 component stop
  port ( clk, reset : in std_logic;
    sdata : in std_logic;
    pdata : buffer std_logic_vector (6 downto 0) );
 end component;
-- hold the letter in "memory"
 component pdatahold
  port (  clk, reset : in std_logic;
    letterin : in std_logic_vector (6 downto 0);
    letterout : buffer std_logic_vector (6 downto 0) );
 end component;
-- find the correct letter
 component letters
  port (  clk, reset : in std_logic;
    letter : in std_logic_vector (6 downto 0);
    getnextletter : buffer std_logic;
    count : buffer std_logic_vector (4 downto 0) );
 end component;
-- sound out the morse code for the appropriate letter
 component soundout
  port (  clk, reset : in std_logic;
    count : in std_logic_vector (4 downto 0);
    sound, letterclk : buffer std_logic );
 end component;
```

```
begin
 serpar1 : stop port map (clkkey, reset, letterser, letterpar);
 parhold : pdatahold port map (getnextletter, reset, letterpar, let);
 decod : letters port map (letterclk, reset, let, getnextletter, count);
 time : soundout port map (clk, reset, count, sound, letterclk);
end ltos;
-- Serial to parallel conversion
  library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity stop is
 port ( clk, reset : in std_logic;
   sdata : in std_logic;
   pdata : buffer std_logic_vector (6 downto 0) );
end stop;
architecture serpar of stop is
 signal count : std_logic_vector (3 downto 0);
 signal preg : std_logic_vector (7 downto 0);
 signal loaddata_l : std_logic;
begin
 process (clk)
begin
 if clk 'EVENT and clk = '0' then
  if reset = '1' then
   count <= (others => '0');
   preg(0) <= sdata; -- start bit
   pzero : for i in 1 to 6 loop
    preg(i) <= '0';
   end loop;
   pzero0 : for i in 0 to 6 loop
    pdata(i) <= '0';
   end loop;
   loaddata_l <= '0';
  elsif loaddata_l = '0' then
   count <= (others => '0');
   loaddata_l <= '1';
   genbits : for i in 0 to 6 loop
    pdata(i) <= preg(i+1);
   end loop;
   preg(0) <= sdata; -- start bit
```

```
  else
   if count(3) = '0' and count(2) = '0' and count(1) = '0' and count(0) = '0' then
    preg(0) <= sdata; -- begin data?? start bit
   elsif count(3) = '0' and count(2) = '0' and count(1) = '0' and count(0) = '1' then
    preg(1) <= sdata; -- begin data
   elsif count(3) = '0' and count(2) = '0' and count(1) = '1' and count(0) = '0' then
    preg(2) <= sdata;
   elsif count(3) = '0' and count(2) = '0' and count(1) = '1' and count(0) = '1' then
    preg(3) <= sdata;
   elsif count(3) = '0' and count(2) = '1' and count(1) = '0' and count(0) = '0' then
    preg(4) <= sdata;
   elsif count(3) = '0' and count(2) = '1' and count(1) = '0' and count(0) = '1' then
    preg(5) <= sdata;
   elsif count(3) = '0' and count(2) = '1' and count(1) = '1' and count(0) = '0' then
    preg(6) <= sdata; -- end data??
   elsif count(3) = '0' and count(2) = '1' and count(1) = '1' and count(0) = '1' then
    preg(7) <= sdata; -- end of data for letter
   elsif count(3) = '1' and count(2) = '0' and count(1) = '0' and count(0) = '1' then
    -- parity
   else
  -- end bit
   end if;
   count <= count + 1;
   loaddata_l <= not( count(3) and (not count(2)) and count(1) and (not count(0)) );
  end if;
 end if;
end process;
end serpar;
library ieee;
use ieee.std_logic_1164.all;
entity pdatahold is
 port ( clk, reset : in std_logic;
   letterin : in std_logic_vector (6 downto 0);
   letterout : buffer std_logic_vector (6 downto 0) );
end pdatahold;
architecture hold of pdatahold is
begin
 process (clk)
 begin
  if clk 'EVENT and clk = '1' then
```

```
   if reset = '1' then
    letterout <= (others => '0');
   else
    pass : for i in 0 to 6 loop
     letterout(i) <= letterin(i);
    end loop;
   end if;
  end if;
 end process;
end hold;
  -- Selects Entered Letter and Outputs appropriate Morse Code
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity letters is
port ( clk, reset : in std_logic;
   letter : in std_logic_vector (6 downto 0);
   getnextletter : buffer std_logic;
   count : buffer std_logic_vector (4 downto 0) );
end letters;
architecture define of letters is
 signal morse : std_logic_vector (3 downto 0);
 constant short : std_logic_vector (4 downto 0) := "11000";
 constant long : std_logic_vector (4 downto 0) := "00111";
begin
 process (clk)
 begin
  if clk 'EVENT and clk = '1' then
   if reset = '1' then
    count <= (others => '0');
    getnextletter <= '1';
    morse <= (others => '0');
   else
-- A    if letter = "0011100" then -- A
     if morse = "0000" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0001" then
      count <= long;
```

```
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
  -- B  elsif letter = "0110010" then -- B
       if morse = "0000" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0011" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
  -- C  elsif letter = "0100001" then -- C
       if morse = "0000" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= long;
```

```
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0011" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- D   elsif letter = "0100011" then -- D
       if morse = "0000" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- E   elsif letter = "0100100" then -- E
       if morse = "0000" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- F elsif letter = "0101011" then -- F
```

```
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- G  elsif letter = "0110100" then -- G
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- H  elsif letter = "0110011" then -- H
```

```
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- I  elsif letter = "1000011" then -- I
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- J  elsif letter = "0111011" then -- J
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
```

```
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- K  elsif letter = "1000010" then -- K
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- L   elsif letter = "1001011" then -- L
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
```

```
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- M elsif letter = "0111010" then -- M
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- N  elsif letter = "0110001" then -- N
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
```

```
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
-- O   elsif letter = "1000100" then -- O
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
-- P  elsif letter = "1001101" then -- P
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
```

```
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- Q  elsif letter = "0010101" then -- Q
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0011" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
  -- R elsif letter = "0101101" then -- R
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
```

```
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
-- S elsif letter = "0011011" then -- S
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0010" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
-- T  elsif letter = "0101100" then -- T
      if morse = "0000" then
       count <= long;
       morse <= morse + 1;
       getnextletter <= '0';
      else
       count <= (others => '0');
       morse <= (others => '0');
       getnextletter <= '1';
      end if;
-- U elsif letter = "0111100" then -- U
      if morse = "0000" then
       count <= short;
       morse <= morse + 1;
       getnextletter <= '0';
      elsif morse = "0001" then
       count <= short;
```

```
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- V   elsif letter = "0101010" then -- V
       if morse = "0000" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0011" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- W  elsif letter = "0011101" then -- W
       if morse = "0000" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= long;
```

```
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
  -- X  elsif letter = "0100010" then -- X
       if morse = "0000" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0011" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
  -- Y  elsif letter = "0110101" then -- Y
       if morse = "0000" then
        count <= long;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
```

```
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0010" then
      count <= long;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0011" then
      count <= long;
      morse <= morse + 1;
      getnextletter <= '0';
     else
      count <= (others => '0');
      morse <= (others => '0');
      getnextletter <= '1';
     end if;
-- Z  elsif letter = "0011010" then -- Z
     if morse = "0000" then
      count <= long;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0001" then
      count <= long;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0010" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0011" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     else
      count <= (others => '0');
      morse <= (others => '0');
      getnextletter <= '1';
     end if;
-- Space
    elsif letter = "01010010" then -- Space
     if morse = "0000" then
```

```
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0010" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0011" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0100" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0101" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0110" then
        count <= (others => '0');
        morse <= morse + 1;
        getnextletter <= '0';
       else
        count <= (others => '0');
        morse <= (others => '0');
        getnextletter <= '1';
       end if;
   -- backspace
      elsif letter = "1100110" then -- backspace
       if morse = "0000" then
        count <= short;
        morse <= morse + 1;
        getnextletter <= '0';
       elsif morse = "0001" then
        count <= short;
```

```
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0010" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0011" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0100" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0101" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0110" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     elsif morse = "0111" then
      count <= short;
      morse <= morse + 1;
      getnextletter <= '0';
     else
      count <= (others => '0');
      morse <= (others => '0');
      getnextletter <= '1';
     end if;
    else
     count <= (others => '0');
     getnextletter <= '1';
    end if;
   end if;
  end if;
 end process;
end define;
library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity soundout is
 port (  clk, reset : in std_logic;
   count : in std_logic_vector (4 downto 0);
   sound, letterclk : buffer std_logic );
end soundout;
architecture timer of soundout is
 type state_type is (idle, determine, pause, short, long);
 signal noise : state_type;
 signal timedelay : std_logic_vector (1 downto 0);
 signal cnt : std_logic_vector (4 downto 0);
begin
 process (clk)
 begin
  if clk 'EVENT and clk = '1' then
   if reset = '1' then
    sound <= '0';
    noise <= idle;
    cnt <= (others => '0');
    letterclk <= '0';
    timedelay <= (others => '0');
   else
    case noise is
     when idle =>
      sound <= '0';
      cnt <= count;
      letterclk <= '0';
      noise <= pause;
      timedelay <= (others => '0');
     when pause =>
      if timedelay = "11" then
       sound <= '0';
       letterclk <= '0';
       noise <= determine;
       timedelay <= (others => '0');
      else
       timedelay <= timedelay + 1;
       noise <= pause;
      end if;
```

```
   when determine =>
    if cnt >= 8 then
     letterclk <= '0';
     noise <= short;
    elsif cnt < 8 and cnt > 0 then
     letterclk <= '0';
     noise <= long;
    else
     noise <= idle;
     sound <= '0';
     letterclk <= '1';
     cnt <= (others => '0');
    end if;
   when short =>
    if cnt > 0 then
     letterclk <= '0';
     sound <= '1';
     cnt <= cnt + 1;
     noise <= short;
    else
     noise <= idle;
     letterclk <= '1';
     sound <= '0';
     cnt <= (others => '0');
    end if;
   when long =>
    if cnt > 0 then
     sound <= '1';
     letterclk <= '0';
     cnt <= cnt + 1;
     noise <= long;
    else
     noise <= idle;
     sound <= '0';
     letterclk <= '1';
     cnt <= (others => '0');
    end if;
  end case;
 end if;
end if; end process; end timer;
```