

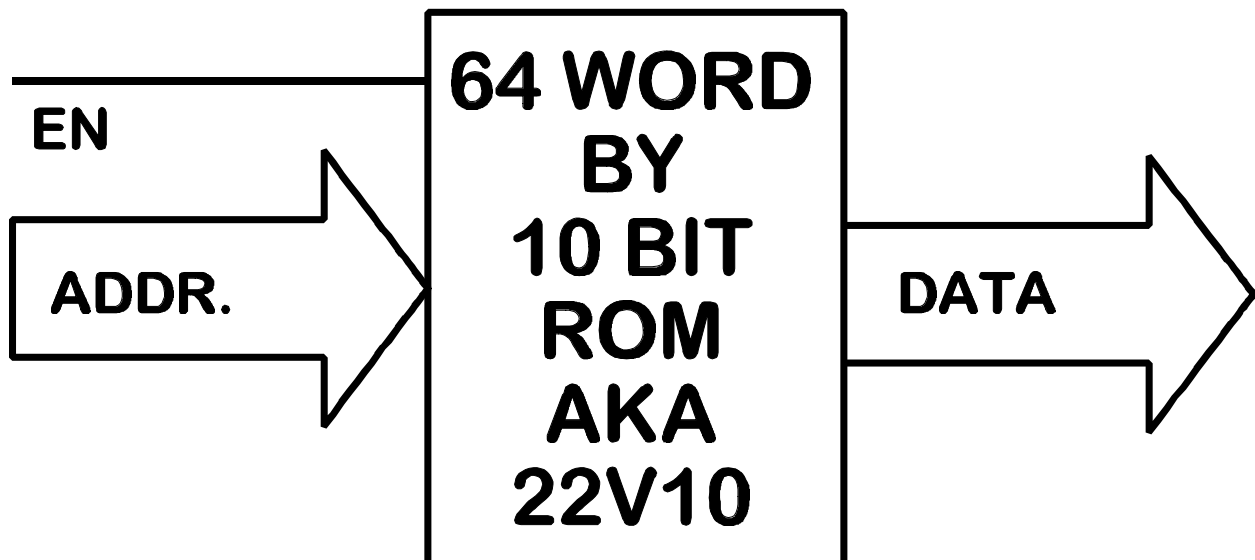
DESCRIBING (AND PROGRAMMING) A PROM IN VHDL

```

library ieee;
use ieee.std_logic_1164.all;
entity rom is port(
  en:      in    bit;
  addr:    in    integer(0 to 63);
  romout:  out   std_logic_vector(9 downto 0)
);
end rom;

architecture rom_data of rom is
  type rom_table is array(0 to 63) of std_logic_vector(9 downto 0);
  constant d:rom_table :=(
    "0000000010", "0010000100", "0100001000", "0110001101",
    "1000010000", "1010010110", "1100011000", "1110011100",
    "1110111100", "1100011000", "1010010100", "1000010001",
    "0110001100", "0100001000", "0011000110", "0000000000",
    "0000000101", "0100001100", "1000010100", "1100011101",
    "1110011000", "1011010010", "0110001000", "0010000000",
    "0000000000", "0000000001", "1110111100", "1110011100",
    "1110011100", "1110011100", "0000000000", "0000000000",
    "0000000010", "0010000100", "0100001000", "0110001101",
    "1000010000", "1010010110", "1100011000", "1110011100",
    "1110111100", "1100011000", "1010010100", "1000010001",
    "0110001100", "0100001000", "0011000110", "0000000000",
    "0000000101", "0100001100", "1000010100", "1100011101",
    "1110011000", "1011010010", "0110001000", "0010000000",
    "0000000000", "0000000001", "1110111100", "1110011100",
    "1110011100", "1110011100", "0000000000", "0000000000" );
begin
  process(addr, en) begin
    if (en='1') then romout<=d(addr);
    else romout<="ZZZZZZZZZZ";
    end if;
  end process;
end rom_data;

```



Describing the ROM data in binary is not usually the easiest way of programming it. Below are two variations.

```

library ieee;
use ieee.std_logic_1164.all;
entity rom is port(
  en:      in      bit;
  addr:    in      integer(0 to 63);
  romout:  out     std_logic_vector(7 downto 0)
);
end rom;

architecture rom_data of rom is
  type rom_table is array(0 to 63) of std_logic_vector(7 downto 0);
  constant d:rom_table :=(
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18",
    X"08",X"21",X"42",X"63",X"F3",X"52",X"31",X"18" );
  begin
  process(addr, en) begin
    if (en='1') then romout<=d(addr); else romout<="ZZZZZZZZ"; end if;
  end process;
end rom_data;

```

The above HEX format is a bit easier to enter. If ASCII characters are to be stored,

```

library ieee;
use ieee.std_logic_1164.all;
entity rom is port(
  addr:    in      integer(0 to 63);
  romout:  out     CHARACTER
);
end rom;

architecture rom_data of rom is
  type rom_table is array(0 to 71) of character;
  constant d:rom_table :=(
    'T','H','I','S',' ','I','S',' ',' ',
    'A',' ','T','E','S','T',' ','0',
    'F',' ','P','R','O','G','R','A',
    'M','M','I','N','G',' ','A','R',
    'O','M',' ','I','N',' ','V','H',
    'D','L',' ','A','N','D',' ','A',
    'L','S','O',' ','M','Y','P','A',
    'T','I','E','N','C','E',' ','N',
    'T','Y','P','I','I','N','G','!'
  );
  begin
  romout<=d(addr);
end rom_data;

```

Note that this last example will compile but produces an error message in the fitter. Perhaps some other conversion library is needed. In any case, if a real ROM is to be programmed, a fitter would not be used. Some ROM software would be used for this.