```
function hdb3_code = HDB3_code(bits, N_sample)
    L = length(bits); % Obtain the length of symbol

    % Generate the AMI code
    AMI = zeros(1, L);
    number1 = 0; % Record the number of 1 voltage
    number0 = 0; % Record the number of 0 voltage
    for i = 1:L
        if bits(i) == 1  % If the symbol is 1, AMI is 1/-1
            number1 = number1 + 1;
            if mod(number1, 2) == 0 % If the symbol is even 1 in the
 sequence, AMI is 1
                AMI(i) = 1;
            else % If the symbol is odd 1 in the sequence, AMI is -1
                AMI(i) = -1;
            end
        else % If the symbol is 0, AMI is 0
            number0 = number0 + 1;
            AMI(i) = 0;
        end
    end
    % Display the AMI code in convenience to verify
    disp('AMI')
    disp(AMI)

    % Generate the step 1 code(check continuous 0s and replace by
 000V)
    hdb3_code = abs(AMI);
    count0 = 0; % Record the number of successive 0
    for i = 1:L
        if abs(AMI(i)) == 1 % If AMI is 1, recount count0
            count0 = 0;
        else % If AMI is 0, accumulatively add count0
            count0 = count0 + 1;
            if count0 == 4 % If count0 is 4, replace 0000 by 000V
                hdb3_code(i) = 2; % 2 represents V
                count0 = 0; % Reset count0
            end
        end
    end
    % Display the step 1 code in convenience to verify
    disp('hdb3_code')
    disp(hdb3_code)

    % Generate the step 1 code(check the number of
    % 1 between the adjacent V to replace 000V by B00V)
    position_v = []; % Record the postion of V in the symbol sequence
    for i = 1:L
        if hdb3_code(i) == 2
            position_v = [position_v, i];
        end
    end
```

```
    disp(position_v) % Display the position of V
    for i = 1:(length(position_v) - 1)
        number1 = 0; % Record the number of 1 between two adjacent V
        for j = position_v(i):position_v(i+1)
            if hdb3_code(j) == 1
                number1 = number1 + 1;
            end
        end
        if mod(number1, 2) == 0 % If even 1, replace 000V by B00V
            hdb3_code(position_v(i+1)-3) = 3; % 3 represents B
        end
    end
    % Display the step 2 code in convenience to verify
    disp('hdb3_code')
    disp(hdb3_code)

    % Generate the step 1 code(determine the polarity)
    for i = 1:L % Set the first 1 voltage to -1
        if hdb3_code(i) == 1
            hdb3_code(i) = -1;
            break;
        end
    end
    sign1 = 0; % Record the current polarity of 1 and B voltage, 0
 represent +
    signv = 0; % Record the current polarity of V voltage, 0 represent
 -
    for i = 1:L
        if (hdb3_code(i) == 1) || (hdb3_code(i) == 3)
            if sign1 == 0 % 0 represent +
                hdb3_code(i) = 1; % May replace 3 by 1
            else % ~0 represent -
                hdb3_code(i) = -1; % May replace -3 by -1
            end
            sign1 = ~sign1; % Invert the polarity of 1 and B voltage
        elseif abs(hdb3_code(i)) == 2
            signv = sign1; % V has the same polarity as that of the
 first previous non-zero code
            if signv == 0 % 0 represent -
                hdb3_code(i) = -1; % Replace -2 by -1
            else % ~0 represent +
                hdb3_code(i) = 1; % Replace 2 by 1
            end
            signv = ~signv; % Invert the polarity of V voltage
        end
    end
    % Display the HDB3 code in convenience to verify
    disp('hdb3_code')
    disp(hdb3_code)

    % Supply the sample point
    hdb3_code = kron(hdb3_code, ones(1, N_sample));
end
```

```
##########

## HDB3_code (# 2 #)
    L = length(bits); % Obtain the length of symbol
```

Published with MATLAB® R2021a