# Gray Code and Error Detection Code

# The Gray Code

- The Gray code is un-weighted and is not an arithmetic code.
- There are no specific weights assigned to the bit positions.
- Important: the Gray code exhibits only a single bit change from one code word to the next in sequence.

# The Gray Code

- Binary-to-Gray code conversion
  - The MSB in the Gray code is the same as corresponding MSB in the binary number.
  - Going from left to right, add each adjacent pair of binary code bits to get the next Gray code bit. Discard carries.

**ex:** **convert** $10110_2$ to Gray code

1 + 0 + 1 + 1 + 0 binary

1 1 1 0 1 Gray

# Gray Code

## Gray Code

Gray code is largely used for getting only single bit change while moving from one consecutive number to the next number. The code is named after Bell laboratories scientist Frank Gray who patented the code in 1953. The table shows the comparative bit change between binary number and its gray code, this comparison will give a clear picture depicting the advantage of having gray code in some applications.

Table
(Gray Code)

| Decimal | Binary | Gray Code |
|---|---|---|
| 0 | 0000 | 0000 |
| 1 | 0001 | 0001 |
| 2 | 0010 | 0011 |
| 3 | 0011 | 0010 |
| 4 | 0100 | 0110 |
| 5 | 0101 | 0111 |
| 6 | 0110 | 0101 |
| 7 | 0111 | 0100 |
| 8 | 1000 | 1100 |
| 9 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

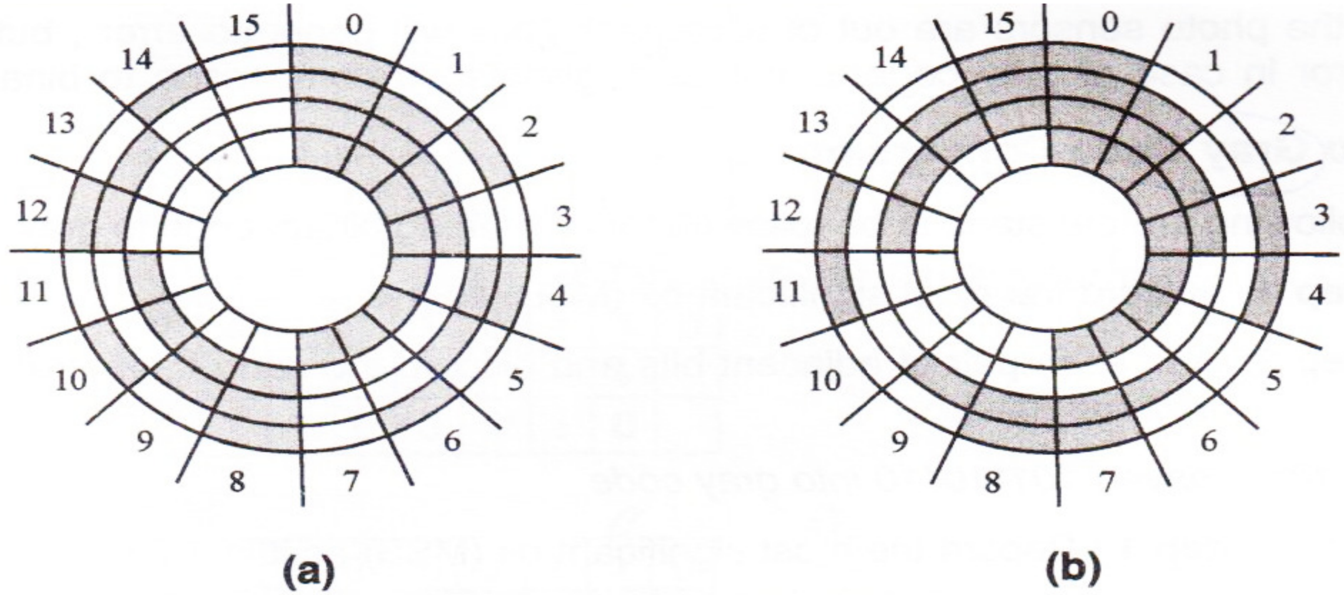


Fig. (a) Binary Encoder Wheel (b) Gray Code Encoder Wheel

***Application:***

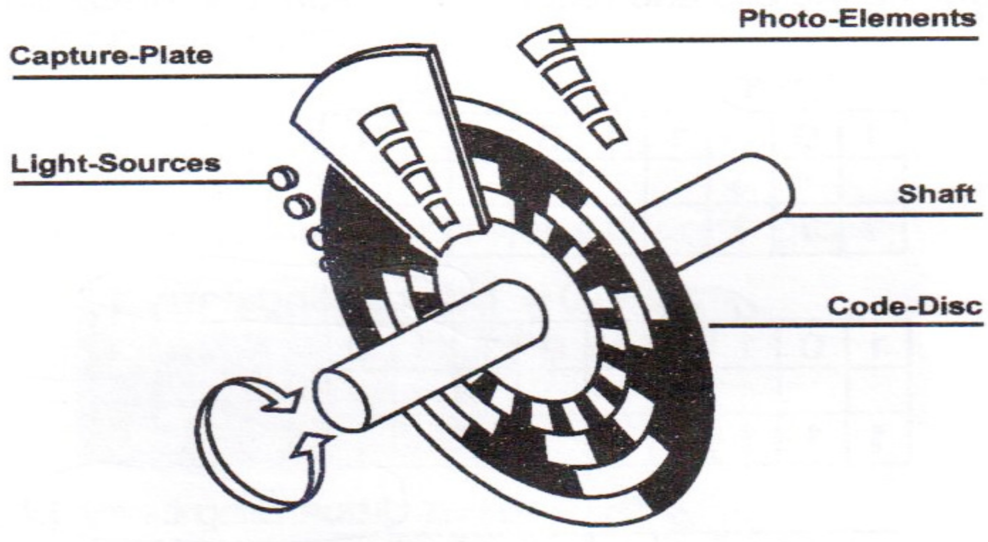


Fig. An Arrangement for Converting Angular Motion into Digital Code (gray Code)

- ## Gray-to-Binary Conversion
  - **The MSB in the binary code is the same as the corresponding bit in the Gray code.**
  - **Add each binary code bit generated to the Gray code bit in the next adjacent position. Discard carries.**

**ex: convert the Gray code word** 11011 to binary

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | Gray |
| | + | + | + | + | |
| 1 | 0 | 0 | 1 | 0 | Binary |

# Even & Odd Parity

- Computers can sometimes make errors during the transmission of data.
- **Even/odd parity:** is basic method for detecting if an odd number of bits has been switched by accident.
- **Odd parity:** The number of 1-bit must add up to an odd number
- **Even parity:** The number of 1-bit must add up to an even number

In even parity the extra bit is inducted in the data to make the number of 1's even

## Even Parity Data Table

| Data | | | | | Parity | Total number of 1's |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 4 |
| 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| 1 | 1 | 0 | 0 | 0 | 0 | 2 |
| 1 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 1 | 0 | 0 | 4 |

# Odd Parity

*Odd Parity*

Odd parity is in contrast to the even parity, here the extra bit is added with the data to make odd sum.

| Data | | | | | Parity | Total number of 1's |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 3 |
| 1 | 1 | 1 | 1 | 0 | 1 | 5 |
| 1 | 0 | 1 | 0 | 1 | 0 | 3 |
| 1 | 0 | 0 | 0 | 1 | 1 | 3 |
| 1 | 1 | 1 | 1 | 1 | 0 | 5 |

From the table it can be very well visualize that 1 bit parity is added to make the number of 1's odd before sending from the transmitter.

- **The computer knows which parity it is using :**
- **If it uses an even parity and If the number of 1-bit add up to an odd number then it knows there was an error.**
- **If it uses an odd: If the number of 1-bit add up to an even number then it knows there was an error.**
- **However, If an even number of 1-bit is flipped the parity will still be the same. But an error occurs**

**Suppose we have an 7-bit binary word (7-digits).**

**If you need to change the parity you need to add 1 (parity bit) to the binary word.**

**You now have 8 digit word.**

**However, the computer knows that the added bit is a parity bit and therefore ignore it.**

## Disadvantage:

The main disadvantage of one dimensional parity is that if noise (external signal) changes data at two places simultaneously then it is not possible to know that an error has occurred in the data. For example, in the third data of odd parity, the 1 has changed to 0 and 0 to 1(shown by bold numbers) then regardless of the erroneous change the number of 1's remains odd , which will not give indication by the checker circuit that error has occurred in the data . To overcome the problem occurring in one dimensional parity the data is sent with two dimensional parity, in this type of process after some number of data the row parity is generated and sent with the data, this reduces some of the problem.

| Data | | | | | Parity | Total number of 1's |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 | 5 |
| 1 | 1 | **0** | 1 | **1** | 1 | 5 |

# Two Dimensional Parity

*Two Dimensional Parity*

| | Data | | | | | Column Parity | Total number of 1's |
|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 | **0** | 1 |
| | 0 | 0 | 1 | 0 | 1 | **1** | 3 |
| | 1 | 1 | 1 | 1 | 0 | **1** | 5 |
| | 1 | 0 | 1 | 0 | 1 | **0** | 3 |
| | 1 | 0 | 0 | 0 | 1 | **1** | 3 |
| | 1 | 1 | 1 | 1 | 1 | **0** | 5 |
| **Row Parity** → | **0** | **1** | **1** | **1** | **1** | **0** | X |
| **Total numbers 1's** → | 5 | 3 | 5 | 3 | 5 | | |

The table above shows two dimensional odd parity. It has column as well as row parity. This row parity is sent at a certain intervals which the receiver knows in advance to match the parities, here the row parity is sent after sixth data. Now if data changes at two places then it will be known by the row parity. Let us carry forward the disadvantage of one dimensional parity in which the data changes at two places simultaneously (see third data).

# Two Dimensional Parity

| | Data | | | | | Column Parity | Total number of 1's |
|---|---|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 1 | 0 | 1 | 1 | 3 |
| | 1 | 1 | 0 | 1 | 1 | 1 | 5 |
| | 1 | 0 | 1 | 0 | 1 | 0 | 3 |
| | 1 | 0 | 0 | 0 | 1 | 1 | 3 |
| | 1 | 1 | 1 | 1 | 1 | 0 | 5 |
| Row Parity → | 0 | 1 | 1 | 1 | 1 | 0 | X |
| | 5 | 3 | 4 | 3 | 6 | | |

Error, even number of 1's

# Error Detection and Correction Code

Hamming code is a error detecting as well as error correcting code named after its inve[ntor] Richard Hamming in 1950, it can detect and correct single-bit error. In other words, t[he] Hamming distance between the transmitted and received code-words must be zero or o[ne] for reliable communication. Alternatively, it can detect and correct single bit error in data

## Code Generation :

Consider a message having four data bits 'D' which is to be transmitted as a 7-bit code word by adding three error control bits. This would be called a (7,4) code.

| | | | | | | |
|---|---|---|---|---|---|---|
| $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |

The three bits to be added are for three EVEN Parity bits (P), where the parity of each is computed as mentioned below as:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $D_7$ | + | $D_5$ | + | $D_3$ | + | $P_1$ | = Even Sum |
| $D_7$ | + | $D_6$ | + | $D_3$ | + | $P_2$ | = Even Sum |
| $D_7$ | + | $D_6$ | + | $D_5$ | + | $P_4$ | = Even Sum |

## Example:

*Construct a seven bit Hamming code with four data bits 1101.*

Putting the data bits at their stipulated places

| $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | | 1 | | |

Now finding the values of the parity bits $P_1$, $P_2$, and $P_4$ by setting even parity sum, for different data bit combinations.

| $D_7$ | + | $D_5$ | + | $D_3$ | + | $P_1$ | = Even Sum |
|---|---|---|---|---|---|---|---|
| 1 | + | 0 | + | 1 | + | 0 | = Even Sum |

| $D_7$ | + | $D_6$ | + | $D_3$ | + | $P_2$ | = Even Sum |
|---|---|---|---|---|---|---|---|
| 1 | + | 1 | + | 1 | + | 1 | = Even Sum |

| $D_7$ | + | $D_6$ | + | $D_5$ | + | $P_4$ | = Even Sum |
|---|---|---|---|---|---|---|---|
| 1 | + | 1 | + | 0 | + | 0 | = Even Sum |

The final seven bit Hamming code for the data 1101 is :

| $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |

# How error is detected and corrected in Hamming Code

The biggest advantage of the Hamming code is, that it has the potential of identifying the location at which the error has occurred as well as correcting it .

To understand the method of identification and correction of the code let us take an example. In the Example -13 we calculated the 7 bit Hamming code for the given data as 1 1 0 0 1 1 0 . We will now creat error at different bit positions and will try to find out the way of identifying and correcting it.

***Error at bit position $D_3$:*** Changing the bit at $D_3$ from 1 to 0. The received data is checked for the even parity for different bit combinations and confirming for even parity.

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | 1 | 0 | 0 | **0** | 1 | 0 | | |
| | 1 | | 0 | | **0** | | 0 | **No** | **7,5,3,1** |
| | 1 | 1 | | | **0** | 1 | | **No** | **7,6,3,2** |
| | 1 | 1 | 0 | 0 | | | | Yes | 7,6,5,4 |

***Error at bit position $D_5$:*** Changing the bit at $D_5$ position and study the effect.

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | 1 | **1** | 0 | 1 | 1 | 0 | | |
| | 1 | | 1 | | 1 | | 0 | **No** | 7,**5**,3,1 |
| | 1 | 1 | | | 1 | 1 | | **Yes** | 7,6,3,2 |
| | 1 | 1 | **1** | 0 | | | | **No** | 7,6,**5**,4 |

The even parity is negative for bit position 7,5,3,1 and 7,6,5,4. Looking at the shaded band, bits at $D_5$ is common which indicates that the data at $D_5$ has been received incorrect and the error is corrected by replacing 0 at that position

***Error at bit position $D_6$:*** The bit has changed from 1 to 0 at sixth bit place ($D_6$).

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | **0** | 0 | 0 | 1 | 1 | 0 | | |
| | 1 | | 0 | | 1 | | 0 | Yes | 7,5,3,1 |
| | 1 | **0** | | | 1 | 1 | | **No** | 7,**6**,3,2 |
| | 1 | **0** | **0** | 0 | | | | **No** | 7,**6**,5,4 |

The even parity is not confirmed for bit combinations 7,6,3,2 and 7,6,5,4. In that band only bit position $D_6$ is common, this indicates the error at that place and the received bit 0 is corrected to 1.

***Error at bit position $D_7$:*** The bit at $D_7$ has changed from 1 to 0 at the receiver end.

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | **0** | 1 | 0 | 0 | 1 | 1 | 0 | | |
| | **0** | | 0 | | 1 | | 0 | **No** | **7,5,3,1** |
| | **0** | 1 | | | 1 | 1 | | **No** | **7,6,3,2** |
| | **0** | 1 | 0 | 0 | | | | **No** | **7,6,5,4** |

The even sum for all three bit combination is not confirmed (shown by shaded portion), only $D_7$ bit is common in all confirmation, this indicates the error at $D_7$, which is corrected by changing the received bit 0 to 1.

***Error at parity bit position $P_1$ :*** If the even parity is not confirmed at any single bit combination out of the three , then the error is in the parity bit . The matrix for error at $P_1$ is shown here.

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | 1 | 0 | 0 | 1 | 1 | **1** | | |
| | **1** | | **0** | | **1** | | **1** | **No** | **7,5,3,1** |
| | 1 | 1 | | | 1 | 1 | | Yes | 7,6,3,2 |
| | 1 | 1 | 0 | 0 | | | | Yes | 7,6,5,4 |

In this matrix the even parity is not confirmed for bit position $P_1, D_3, D_5, D_7$ , this infers error at $P_1$. The matrix for error at $P_2$ and $P_4$ is shown below.

***Error at parity bit position $P_2$ :***

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | 1 | 0 | 0 | 1 | **0** | 0 | | |
| | 1 | | 0 | | 1 | | 0 | Yes | 7,6,3,1 |
| | **1** | **1** | | | **1** | **0** | | **No** | **7,6,3,2** |
| | 1 | 1 | 0 | 0 | | | | Yes | 7,6,5,4 |

| | $D_7$ | $D_6$ | $D_5$ | $P_4$ | $D_3$ | $P_2$ | $P_1$ | Even Parity | Bit Places |
|---|---|---|---|---|---|---|---|---|---|
| Transmitted Data → | 1 | 1 | 0 | 0 | 1 | 1 | 0 | | |
| Received Data → | 1 | 1 | 0 | **1** | 1 | 1 | 0 | | |
| | 1 | | 0 | | 1 | | 0 | Yes | 7,6,3,1 |
| | 1 | 1 | | | 1 | 1 | | Yes | 7,6,3,2 |
| | **1** | **1** | **0** | **1** | | | | **No** | **7,6,5,4** |

From the above narration it is concluded , that when error occurs at any of the data bit confirmation for even parity will fail at two or three confirmation bit combination and whe an error occurs at any of the parity bit (P ) , confirmation for even sum will fail at single pla only.

# ASCII Code

## ASCII Code

The ASCII stands for “American standard code for intonation interchange”. This code is used world wide in computers. There are two formats of ASCII code:

ASCII 7 bit code
ASCII 8 bit code

ASCII 8 bit code is extended version of ASCII 7 bit code . To have an idea about the code ASCII 7 bit code is briefly explained here.

The ASCII 7 bit code is universally adopted by all computer manufacturers to encode 8 characters, 10 numerals, 23 punctuation and other symbols marked on keyboards, video displays etc.

The format for such code is :

$$X_6 \quad X_5 \quad X_4 \qquad X_3 \quad X_2 \quad X_1 \quad X_0$$

Where x represents 0 or 1. The value of $x_6x_5x_4$ $x_3x_2x_1x_0$ can be seen from table

Let us take some examples of ASCII Code

The letter D is coded as

D : 100 0100.

Here $x_6x_5x_4$ for alphabet D is 100 and $x_3x_2x_1x_0$ is 0100. Therefore the ASCII code for D is 100 0100

| $X_3$ | $X_2$ | $X_1$ | $X_0$ | $X_6\,X_5\,X_4$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 010 | 011 | 100 | 101 | 110 | 111 |
| 0 | 0 | 0 | 0 | SP | 0 | @ | P | | p |
| 0 | 0 | 0 | 1 | ! | 1 | A | Q | a | q |
| 0 | 0 | 1 | 0 | “ | 2 | B | R | b | r |
| 0 | 0 | 1 | 1 | # | 3 | C | S | c | s |
| 0 | 1 | 0 | 0 | $ | 4 | D | T | d | t |
| 0 | 1 | 0 | 1 | % | 5 | E | U | e | u |
| 0 | 1 | 1 | 0 | & | 6 | F | V | f | V |
| 0 | 1 | 1 | 1 | , | 7 | G | W | g | W |
| 1 | 0 | 0 | 0 | ( | 8 | H | X | H | X |
| 1 | 0 | 0 | 1 | ) | 9 | I | Y | I | Y |
| 1 | 0 | 1 | 0 | * | : | J | Z | J | z |
| 1 | 0 | 1 | 1 | + | ; | K | | k | |
| 1 | 1 | 0 | 0 | ‘ | < | L | | L | |
| 1 | 1 | 0 | 1 | - | = | M | | M | |
| 1 | 1 | 1 | 0 | . | > | N | | N | |
| 1 | 1 | 1 | 1 | / | ? | O | | o | |

**Example** : ***Write down the output of an ASCII keyboard when you type :***
*WEL - COME.*

***Solution:*** From the table , writing down the ASCII Codes.

W: 101 0111
E: 100. 0101
L: 100 1100
-: 010 1101
C: 100 0011
0: 100 1111
M : 100 1101
E: 100 0101

# Extended Binary Coded Decimal Interchange Code

EBCDIC (Extended Binary Coded Decimal Interchange Code) is a character encoding set used by IBM mainframes . EBCDIC uses the full 8 bits available to it, EBCDIC has a wider range of control characters than ASCII.

A single EBCDIC byte occupies eight bits, which are divided in two halves or nibbles. The first four bits is called the zone and represent the category of the character, whereas the last four bits is called the digit and identify the specific character. The EBCDIC code table is shown below.

**EBCDIC Code Table**

| Char | EBCDIC | HEX | Char | EBCDIC | HEX | Char | EBCDIC | HEX |
|---|---|---|---|---|---|---|---|---|
| A | 100 0001 | 41 | P | 1101 0111 | D7 | 4 | 1111 0100 | F4 |
| B | 1100 0010 | C2 | Q | 1101 1000 | D8 | 5 | 1111 0101 | F5 |
| C | 1100 0010 | C3 | R | 1101 1001 | D9 | 6 | 1111 0110 | F6 |
| D | 1100 0100 | C4 | S | 1110 0010 | E2 | 7 | 1111 0111 | F7 |
| E | 1100 0101 | C5 | T | 1110 0011 | E3 | 8 | 1111 1000 | F8 |
| F | 1100 0110 | C6 | U | 1110 0100 | E4 | 9 | 1111 1001 | F9 |
| G | 1100 0111 | C7 | V | 1110 0101 | E5 | | | |
| H | 1100 1000 | C8 | W | 1110 0110 | E6 | | | |
| I | 1100 1001 | C9 | X | 1110 0111 | E7 | | | |
| J | 1101 0001 | D1 | Y | 1110 1000 | E8 | | | |
| K | 1101 0010 | D2 | Z | 1110 1001 | E9 | | | |
| L | 1101 0011 | D3 | 0 | 1111 0000 | F0 | | | |
| M | 1101 0100 | D4 | 1 | 1111 0001 | F1 | | | |
| N | 1101 0101 | D5 | 2 | 1111 0010 | F2 | | | |
| O | 1101 0110 | D6 | 3 | 1111 0011 | F3 | | | |