MSX Article

MARMSX

Screen 2 Layout

Summary

In this article we will discuss on how the MSX screen 2 works and present some stuff that can be made on this screen.

1- The Screen 2 Layout

MSX screen 2 is composed by 768 pixels characters with size of 8x8 pixels, forming 32 characters per row and 24 per column. Figure 1.1 shows the screen 2 character layout.

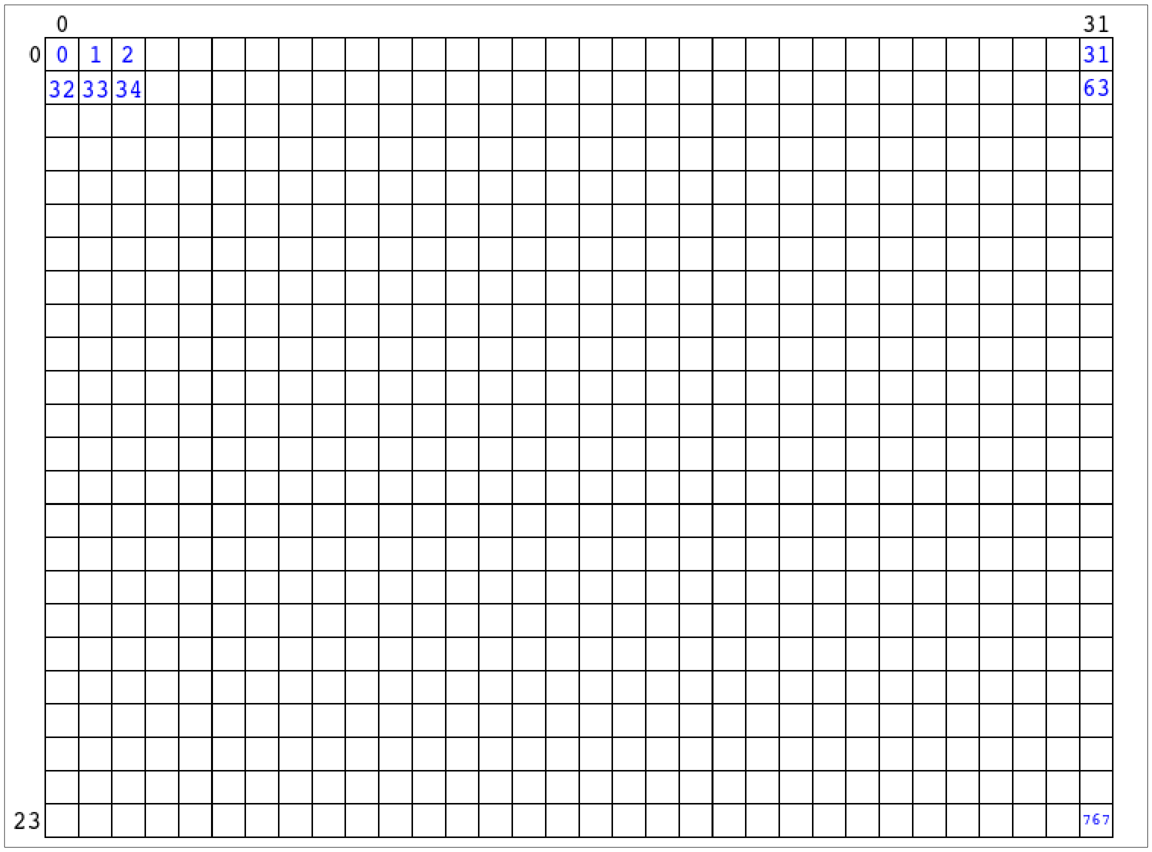


Figure 1.1. Screen 2 division by 8x8 characters.

The sequence of characters in memory is presented by the numbers in blue in the figure above.

Each character is divided into lines, where each line is controlled individually. Unfortunately, screen 2 does not control pixels individually which may sometimes results on some artifacts on screen. Only MSX 2 screens 5 to 8 are able to control pixels individually.

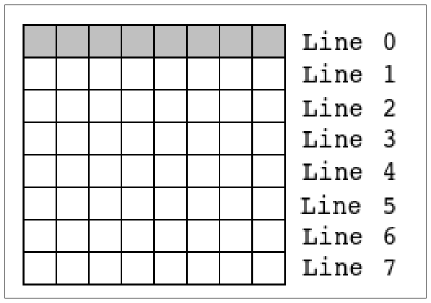


Figure 1.2. Character division by lines.

Each character is generated by two tables located in the MSX video memory (VRAM): the pattern table and the color table. The color table is responsible for setting the front and background colors for each 1x8 pixels line. The pattern table is responsible for the pixels colors pattern, according to the colors defined for that line.

For example, let's define the line 0 of a given character with the MSX 1 colors indexes 10 and 12, using the pattern shown in the figure 1.3.

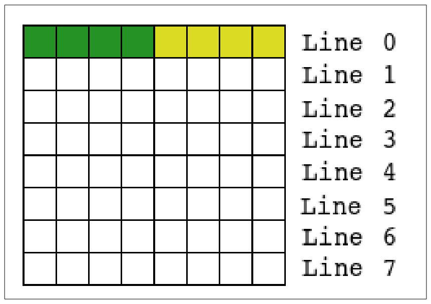


Figure 1.3. Configuring a character's line.

Each character's line is controlled by the same relative position in both pattern and color tables. In the color table, one byte defines the front and the background color. The most significant bits control the front color, while the least significant bits control the background color.

	Front color				Background color			
Bit	7	6	5	4	3	2	1	0

In order to set the line green and yellow like in the figure 1.3, we may define the front color as 10 (&B1100) and the background color as 12 (&B1100). We can also set 12 as front color and 10 as background color, but we must keep the correspondence between the color and pattern tables for that line.

Each bit in the pattern table controls one pixel of the character line. Value 0 means that the selected color is the background color, while value 1 means the front color.

Once we have defined the color index 10 for the front color and 12 for the background color, the value 0 means that the pixel will turn into color 12, while value 1 will turn the pixel into color 10. According to that, the byte configuration for the line 0 (figure 1.3) is the following: &B00001111 (or &HF or 15).

We need more one thing to be able to reproduce this line on the MSX screen 2: to find out how to calculate the address of a line in both pattern and color tables.

Suppose that we want to change the line 4 of the character number 34 (see figure 1.1). This character is located at row 1 and column 2. We already know that each line takes 1 byte in each table and that each character takes 8 bytes. So, if we know the sequence of this character on the screen (it is 34), we multiply it by 8 and add the line number that we want to change.

If we do not know the character sequence number, we may use the character screen coordinates to calculate it.

$$pos\_char = row \times 32 + col$$

In our example, the character is located at row 1 and column 2: 1x32 + 2 = 34.

Remember that each character from pattern and color tables are connected by their locations. We cannot mix one character from the pattern table with another from the color table. According to that, the general formula to calculate the line address in the pattern and color tables is:

$$add = table\_start + pos\_char \times 8 + char\_row$$

Where table_start is the initial address for each table in the VRAM:

```
Initial address for the color table: 8192 or &H2000
Initial address for the pattern table : 0 or &H0000
```

We are now able to draw the line of the figure 1.3 at position 34 on MSX.

```
10 SCREEN 2
20 VPOKE &H2000 + 34*8 + 4, &B10101100 ' Control line color
30 VPOKE 0 + 34*8 + 4, &B00001111 ' Control line pattern
40 GOTO 40
```

Keep in mind that the pattern and color tables define a set of 8x8 characters, and the resulting figure is a great mosaic of 768 characters.

1.1. The name table

After defining the shape and colors of the characters, we can control which and where they will appear on the screen using the name table. With this table, we can create new and many mosaics using the 8x8 characters from pattern and color tables.

The name table has the same layout shown in figure 1.1 and it defines the correspondence between a physical position on the screen and a character from the pattern and color tables. Now, each character is represented by one byte. The position in the name table (table index) defines the physical location on the screen and the value of it defines the character from the pattern and color tables.

But, life is not easy. The screen is vertically divided into three areas with the same size, and we can only access the characters from the pattern and color tables located at the same physical areas. According to that, each area holds 256 characters numbered from 0 to 255.

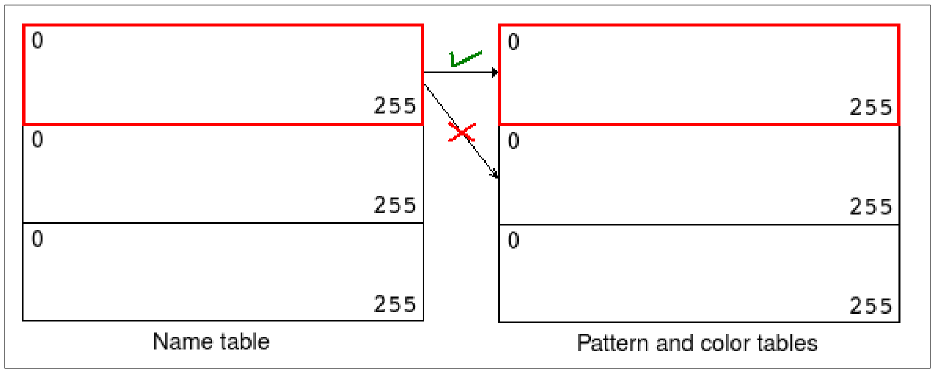


Figure 1.4. Area correspondence between name table and pattern/color tables.

The next example will show how the name table works. Four characters from the Brazilian game *Alcatraz, a Fuga Impossivel* [1] are used.

```
10 COLOR 15,4,4:SCREEN 2
20 FOR E=0 TO 31
30 READ A$ : VPOKE E,VAL("&H"+A$)
40 NEXT E
50 FOR E=0 TO 31
60 READ A$ : VPOKE &H2000+E,VAL("&H"+A$)
70 NEXT E
80 GOTO 80
1000 ' Pattern table
1010 DATA 00,00,00,00,00,00,00,00
1020 DATA 01,01,01,FF,10,10,10,FF
1030 DATA 18,18,18,18,18,18,18,18
1040 DATA 18,18,3C,5A,99,18,24,24
1050 ' Color table
1060 DATA 00,00,00,00,00,00,00,00
1070 DATA E6,E6,E6,E6,E6,E6,E6,E6
1080 DATA 10,40,10,40,10,40,10,40
1090 DATA 10,D0,10,10,10,10,10,10
```

The program changes the pattern and color tables in order to draw the following characters on the screen:

- Empty – code 0
- Brick – code 1
- Bars – code 2
- Police – code 3

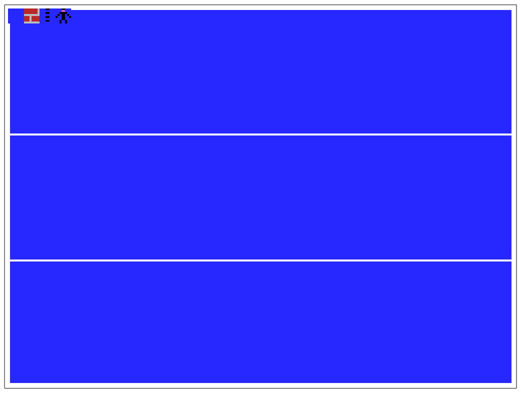


Figure 1.5. Creating characters in pattern/color tables.

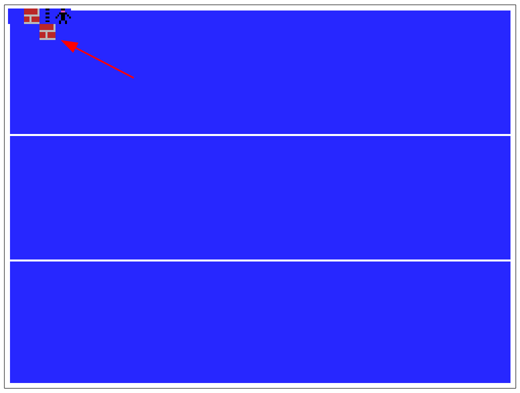
The name table is always initialized from 0 to 255, making all characters visible on screen.

The start address of name table is 6144 or &H1800. Thus, each character now takes one byte instead of 8. So, we rewrite the pattern and color tables' address formula as follows:

$$add = table\_start + part \times 256 + chars\_pos$$

Let's change character number 34 on the screen (see figure 1.1) to display the brick (character code 1). For that, add the following line in the program:

```
75 VPOKE 6144 + 0 + 34, 1
```



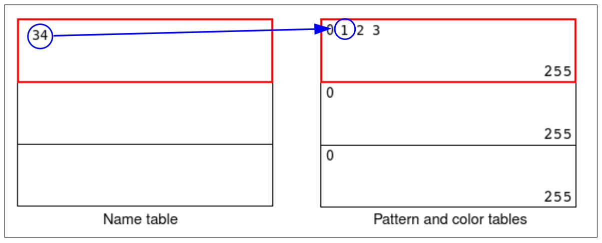


Figure 1.6. Adding a brick at position 34.

As we can see in figure 1.6, the name table index 34 makes reference to the character code 1 in the pattern and color tables, both in the first section of the screen.

And if we want to put a brick character on the second section of the screen?

```
75 VPOKE 6144 + 1*256 + 34, 1
```

The brick is not there! Why? Because we are trying to draw a character from the first section of the screen on the second section. In fact, when we access the second section of the name table, we are also referring to the characters from the second section of the pattern and color tables, and all characters there are empty.

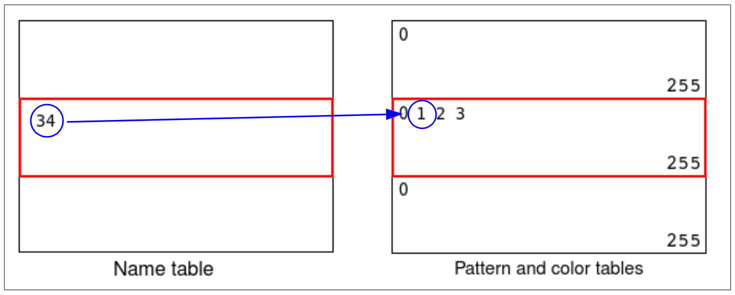


Figure 1.7. Drawing on the second part of the screen.

To solve this issue, we must repeat the characters from the first section of the screen in the the second one. If we want to use these characters in the whole screen, we must repeat them in all sections of the screen.

Add the following lines:

```
35 VPOKE E+2048,VAL("&H"+A$) : VPOKE E+4096,VAL("&H"+A$)
...
65 VPOKE &H2000+E+2048,VAL("&H"+A$) : VPOKE &H2000+E+4096,VAL("&H"+A$)
```

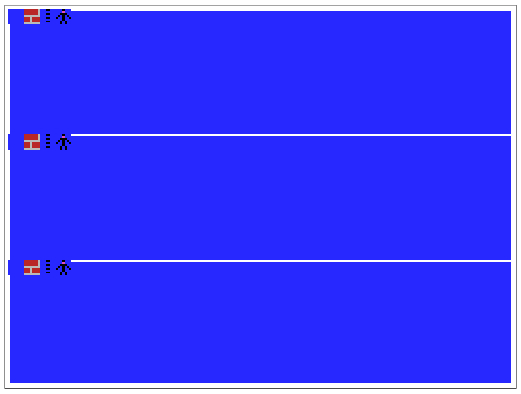


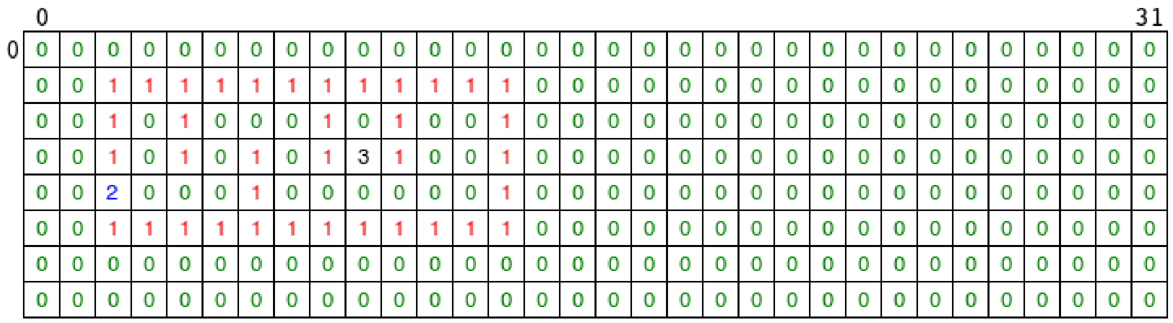
Figure 1.8. Character patterns in all three parts.

Try this code:

```
75 VPOKE 6144 + 1*256 + 34, 1
```

Now the brick is drawn on the screen.

We can create nice stuffs using screen 2 tables: scenarios, big game characters, scrolls etc. For example, we will now draw a maze on the first section of the screen. For that, we must configure the first section of the name table as seen on figure 1.9.



	0																															31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	1	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	1	0	1	0	1	3	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	2	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 1.9. Name table configuration.

In Basic:

```
10 COLOR 15,4,4:SCREEN 2
20 FOR E=0 TO 31
30 READ A$ : VPOKE E,VAL("&H"+A$)
40 NEXT E
50 FOR E=0 TO 31
60 READ A$ : VPOKE &H2000+E,VAL("&H"+A$)
70 NEXT E
80 FOR E=0 TO 255
90 READ A$ : VPOKE &H1800+E,VAL("&H"+A$)
100 NEXT E
120 GOTO 120
1000 ' Pattern table
1010 DATA 00,00,00,00,00,00,00,00
1020 DATA 01,01,01,FF,10,10,10,FF
1030 DATA 18,18,18,18,18,18,18,18
1040 DATA 18,18,3C,5A,99,18,24,24
1050 ' Color table
1060 DATA 00,00,00,00,00,00,00,00
1070 DATA E6,E6,E6,E6,E6,E6,E6,E6
1080 DATA 10,40,10,40,10,40,10,40
1090 DATA 10,D0,10,10,10,10,10,10
1500 ' Name table
1510 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1520 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1530 DATA 0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0
1540 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1550 DATA 0,0,1,0,1,0,0,0,1,0,1,0,0,1,0,0
1560 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1570 DATA 0,0,1,0,1,0,1,0,1,3,1,0,0,1,0,0
1580 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1590 DATA 0,0,2,0,0,0,1,0,0,0,0,0,0,1,0,0
1600 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1610 DATA 0,0,1,1,1,1,1,1,1,1,1,1,1,1,0,0
1620 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1630 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1640 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1650 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1660 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

The result on MSX can be seen in the figure 1.10.

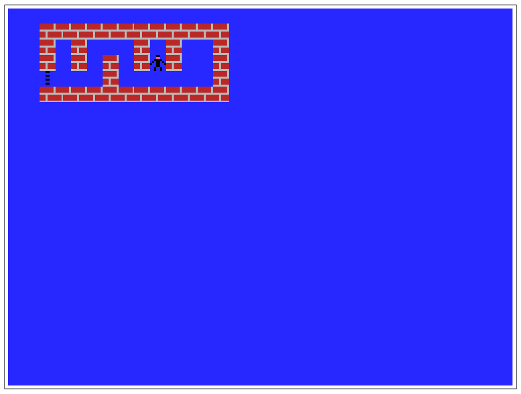


Figure 1.10. Creating scenarios using the name table.

1.2. The "palette effect"

We can generate an interesting effect on screen 2 whenever we set a region (or the whole area) with the same character code. By changing the character's pattern and/or colors, the changes are immediately reflected on the screen.

Let's see the next program.

```
10 COLOR 15,4,4:SCREEN 2:OPEN"GRP:"AS#1
20 FOR E=6144 TO 6399 : VPOKE E,0 : NEXT E
30 PRESET(10,70),0:PRINT#1,"Press enter":A$=INPUT$(1)
40 FOR E=0 TO 7
50 READ A$ : VPOKE E,VAL("&H"+A$)
60 NEXT E
70 FOR E=0 TO 7
80 READ A$ : VPOKE &H2000+E,VAL("&H"+A$)
90 NEXT E
100 GOTO 100
1000 ' Pattern table
1020 DATA 01,01,01,FF,10,10,10,FF
1050 ' Color table
1060 DATA E6,E6,E6,E6,E6,E6,E6,E6
```

As soon as we press the Enter key, we change the character 1 from empty to brick shape in the pattern and color tables, and all the first section of the screen is immediately changed to brick.

Now, let's try the next program.

```
10 COLOR 15,4,4:SCREEN 2:OPEN"GRP:"AS#1
20 FOR E=6144 TO 6399 : VPOKE E,0 : NEXT E
30 FOR E=0 TO 7
40 READ A$ : VPOKE E,VAL("&H"+A$)
50 NEXT E
60 FOR E=0 TO 7
70 READ A$ : VPOKE &H2000+E,VAL("&H"+A$)
80 NEXT E
90 PRESET(10,70),0:PRINT#1,"Press enter":A$=INPUT$(1)
100 FOR E=0 TO 7 : VPOKE &H2000+E,&H46 : NEXT
105 FOR T=1 TO 100 : NEXT T
110 FOR E=0 TO 7 : VPOKE &H2000+E,&HE6 : NEXT
115 FOR T=1 TO 100 : NEXT T
120 GOTO 100
1000 ' Pattern table
1020 DATA 01,01,01,FF,10,10,10,FF
1050 ' Color table
1060 DATA E6,E6,E6,E6,E6,E6,E6,E6
```

In this program, we only change the brick's color table. The color of the cement between the brick is alternated between 4 and 14 (&HE).

1.3. More about screen 2

MarMSX homepage brings more articles on MSX 1 screens layout. For example, the Basic course [2] has the following topics:

- The name, character and color tables.
- Drawing on screens 0 and 1.
- Animation on screen 2.
- Alcatraz game study.

The game creation course [3] deeply discusses on this subject, explaining how to create scenarios, characters and animation. The article Sprites and Gravity [4] and the Assembly course [5] talk about the sprites. At last, the Screen 2 Show project [6] has another article about screen 2 layout.

Credits and References

This article was written by Marcelo Silveira.

Date: January 2021
E-mail: flamar98@hotmail.com
Homepage: http://marmsx.msxall.com

References:

[1] – Jogos de Habilidade, Many Authors. Editora Aleph, Brasil, 1986.
[2] – Basic Course. At: http://marmsx.msxall.com/cursos/basic.
[3] – Games Course in Basic. At: http://marmsx.msxall.com/cursos/basgame.
[4] – Sprites and Gravity, Marcelo Silveira. At: http://marmsx.msxall.com/artigos.
[5] – Assembly Course. At: http://marmsx.msxall.com/cursos/assembly.
[6] – Screen 2 Show, Arquitetura da Screen 2. At: http://marmsx.msxall.com.

* Please, access MarMSX courses in Portuguese using Google Chrome and translate it to your language.