Correlation

The Custom-Font SDK for Ti-83+ Ti-Basic Programs

Rickie Malgren, aka Hot Dog

Table Of Contents

FOR BEGINNERS:

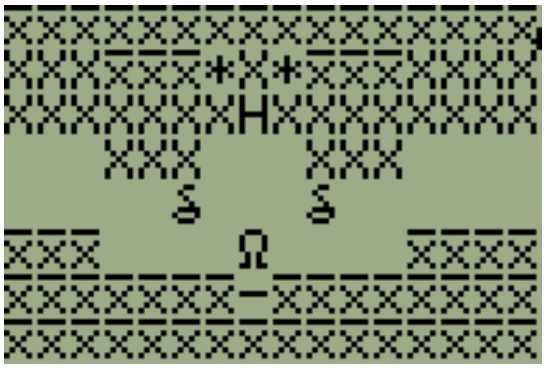
ADVANCED TOPICS:

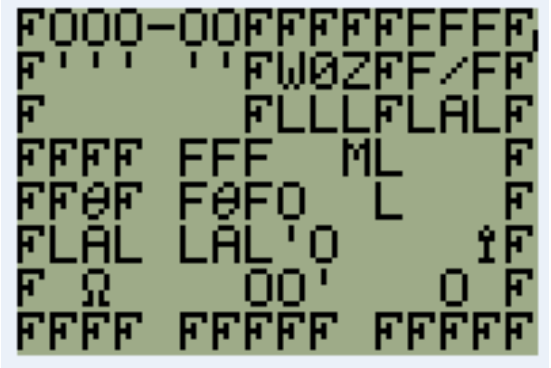
CREDITS:

What is Correlation?

When it comes to the Ti-83+, a lot of games are written in Ti-Basic. Does that mean the games are inferior and undeserving of praise? **ABSOLUTELY NOT!** Some of the greatest games ever written for the Ti-83+ are written in Ti-Basic.

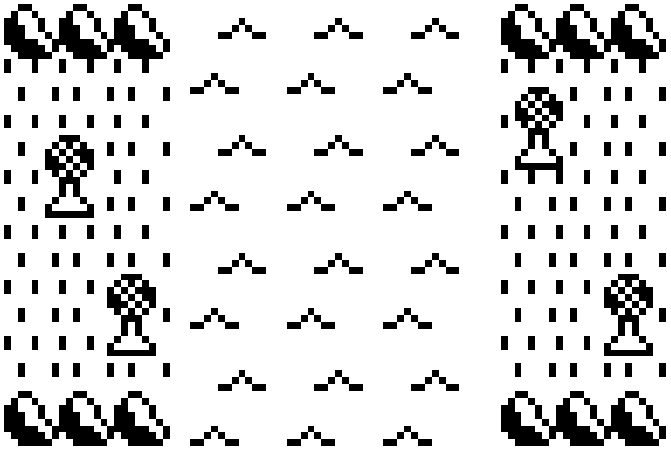
Many of these games are text-based, meaning text is used for graphics. In such games, text can represent rocks, trees, a person, bears, ducks, even water if the programmer wants.





Sadly, this text stuff doesn't fully resemble trees, rocks, walls, people, etc. Do you know why? It's because a text-based game has to use the Ti-83+ font. And the Ti-83+ font contain mostly letters, numbers, symbols and punctuation. It contains no pictures, just text!

Wouldn't it be nice if you could replace or edit the TI-83+ font? What if you could change all the text in the font into pictures so that when you use Output(or Text(you get something like this?



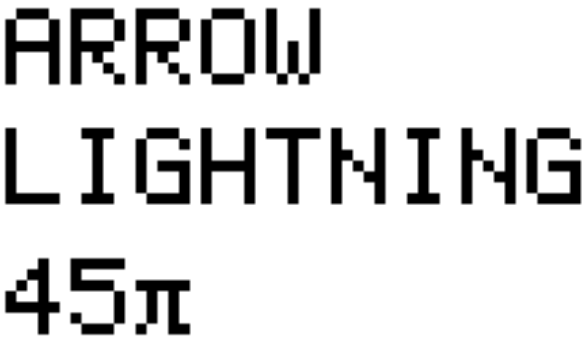
But replacing or editing the Ti-83+ font is hard or impossible, so the next best thing is creating your own font and using it in a Ti-Basic program. And with Correlation, you can do that! You can create a font that uses pictures in place of text. Anything you can do with the Ti-83+ font, you can do with your custom font to create beautiful maps, worlds and game levels.

And best of all, your Ti-Basic program, which uses Correlation, will run faster than a similar Ti-Basic program that does not use Correlation. Hard to believe? You'll definitely see a difference as you write processor-intensive games that use Correlation.

This guide is meant to give you a sure footing in using customized fonts in your Ti-Basic game. So if you're ready, start by looking at the terms on the next page and making sure that you understand what they mean. Or, my friend Rebma Boss prepared a video tutorial on YouTube if you are interested: (LINK COMING SOON!)

Important Terms

Character: A single symbol used in creating sentences. Letters of the alphabet, punctuation marks and numbers are all characters. The Ti-83+ font has a total of 256 characters.



The first line, "ARROW," has 5 characters. The second line has 9 characters. The third line has 3 characters.

It is NOT necessary for your Correlation Font to have 256 characters. You can have as few as 5 characters in your font, or a moderate number of characters such as 110.

ASCII Value: A number that uniquely identifies and defines a particular character. Every character that you can see on your computer screen (or Ti-83+ screen) has a number that distinguishes that character from another character.

<u>Some example ASCII Values:</u>

Uppercase A = 65

0 = 48

& = 38

Lowercase z = 122

Uppercase Letters from A to Z = 65 to 80, one number for each letter

Lowercase Letters from a to z = 97 to 122, one number for each letter

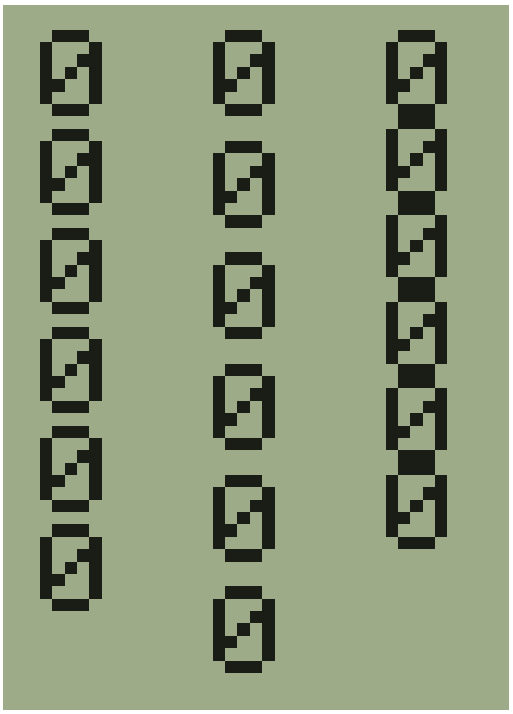
Starting Character Number: The smallest ASCII value allowed by a Correlation font. If you have a starting character number of 68 in your font, you cannot display characters with ASCII values smaller than 68. If the Ti-83+ font had a starting character number of 68, you would not be able to display the sentence ABC using your TI-Basic program.

Ending Character Number: The largest ASCII value allowed by a Correlation font. If you have an ending character number of 96 in your font, you cannot display characters with ASCII values larger than 96. If the Ti-83+ font had an ending character number of 96, you would not be able to display any lowercase letters using your Ti-Basic program.

Horizontal Blank Space: The number of white pixels found between two characters. This space keeps characters from running into each other, as well as making text easy to read.

In the first line of 0s, there is one horizontal blank space between each 0. In the second line of 0s, there are **two** horizontal blank spaces between each 0. In the last line, there are NO horizontal blank spaces between each 0, a good idea for fonts that use graphical images in place of text.

Vertical Blank Space: The number of white pixels counted from the bottom of one character to the top of the next. This space keeps lines from running into each other, as well as making text easy to read.



In the first column of 0s, there is one vertical blank space between each 0. In the second column of 0s, there are **two** vertical blank spaces between each 0. In the last column, there are NO vertical blank spaces between each 0, a good idea for fonts that use graphical images in place of text.

Introduction to Using Custom Fonts in a Ti-Basic Program

As you know, when you want to display text in a Ti-Basic program, you can normally use Output(to display homescreen-style text on 16 columns and 8 rows. You can also use Text(to display text on 96 rows and 64 columns.

Whenever you create a custom font and want to use it in your Ti-Basic program, you can still use Output(and Text(to display text using the TI-83+ font. You will need to use ln(to display homescreen-style text with a custom font, and e^(to display anywhere-on-screen text with a custom font. (By specifying only 1 parameter, you can continue to use ln and e^ for math.)

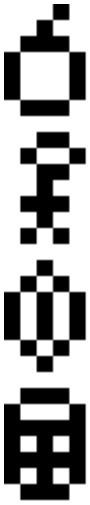
ln(1,1, Str0)

e^(45, 20, sub(Str1,1,2))

Now, I'm really sorry for stating the obvious here, but when you use Output(and Text(, you are telling the calculator exactly what you want it to display on the screen.

Output(1, 1, “ABCFE”)	You will see the string “ABCFE.” You will not see something funky such as 183.!328.
Text(5, 2, “ACORN”)	You will not see the string “DOG,” and you will not see the word “BIRD.” You will see the string “ACORN.”

As you can see, your calculator will display exactly what you tell it to. So let’s say you have this font that you made:



It would be nice if you could type in something like

Output(1,1, “ ◆◆◆◆◆◆ ”). But you can’t! That’s because you’re only able to type characters that are part of the TI-83+ font. This diamond symbol in the font is not part of the Ti-83+ font, so you can’t type it using your keyboard.

However, you **are** able to type in Output(1,1, “AAAAAA”). Can you see where I’m going? Normally, the calculator will display what you tell it do display. But with the help of Correlation, you can tell the calculator to display something DIFFERENT from what you type! In

this case, you can tell the calculator to display the diamond from your font every time it reads an A. So ln(1,1, "AAAAAA") would display 6 diamonds. (By the way, remember that you use either ln or e^ to display text using a custom font.)

And you don't need to stop there! Using Correlation, you could tell the Ti-83+ to display the soda pop can (at the bottom of the sample font) whenever your Ti-83+ reads a "B" inside of a string. So ln(5,7, "ABBBA") would display diamond, soda, soda, soda, diamond.

I'm going to call this process **assigning.** Essentially, you assign a character of your custom font to a character of the Ti-83+ font. In the above examples, you assigned a diamond to the letter "A", and you assigned a soda can to the letter "B".

There's a catch when you assign characters from your font to characters of the Ti-83+ font. You can only assign the **first** character of your font, after which the rest of the characters are assigned automatically, in the order the characters are drawn.

Here's the deal. Taking a look at the example font on page 8, let's say you assign the bomb to a particular character of the Ti-83+ font—we'll refer to this Ti-83+ character as Character 1. With the bomb assigned to Character1, the person in your font will be assigned to the character after Character 1. The diamond in your font will be assigned to the character **Two Characters** away from Character 1. As you might expect, the soda-can will be assigned three characters away from character 1.

So let's say you assign the bomb to the letter "T." Then every time the calculator reads the letter T in a display routine, it will show a bomb. If the calculator reads a "U" inside of a string, it will show a person. The calculator will show a diamond if it reads a V, and a soda can when

it reads a W. ln(1,5, “TTWUV”) translates to displaying bomb, bomb, soda, person, diamond.

So, you’re probably asking by now, “If I know what TI-83+ character to use to display the first character of my font, how will I know what characters of the Ti-83+ font to use to display OTHER characters of my personal font?” The answer is simple: use the table I have given you, called “Ti-83+ Character Codes”. This page provides a bunch of characters you can type, from left to right, top to bottom, in the order they appear in the Ti-83+ font. Since these characters are in order, you can look at what you assigned the first character of your custom font to, and find out other characters from there.

Suppose that you look at the table and assign the “bomb” in your font to the number 8. Then Correlation will automatically assign the person to the number 9. Whenever you use Output(with a negative sign in the string, you will see a diamond instead of a negative sign. And finally, E will be replaced with the soda can.

By the way, be aware that there are some ASCII values on the Ti-83+ that you cannot type characters for. These are marked with black boxes on the table.

Designing a Font

Creating your own font is relatively straight forward, but there are some considerations you have to take into account before you begin the drawing process. You will also need a computer with a program that can create monochrome bitmaps. For computers equipped with Microsoft Windows, I highly recommend Microsoft Paint.

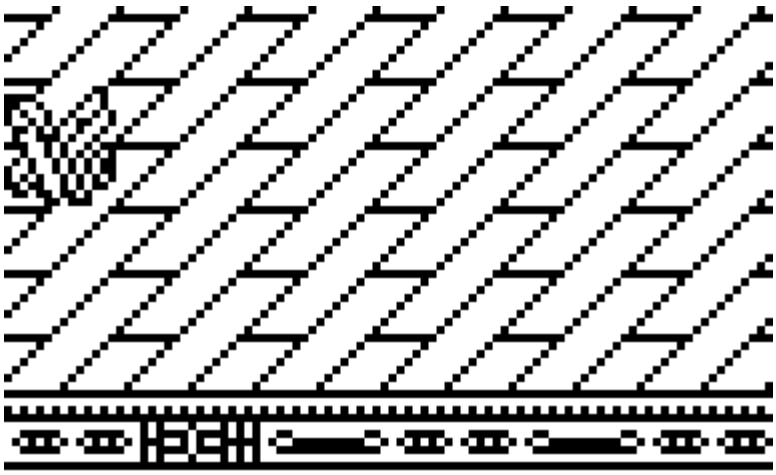
The first thing you want to consider is how big you want each character in your font. The smaller the characters in your font are, the more characters you can display on screen at once. However, small characters aren't as detailed, and require bigger strings in your Ti-Basic program. Larger characters are more detailed and require less string space in your Ti-Basic program, but you can't display as many distinct characters on screen at once.

The standard Ti-83+ Homescreen font has characters 5 pixels wide and 7 pixels high. (I am calling this font a 6x8 font because there's one horizontal blank space and one vertical blank space for each character. This technically means each character needs its own space 6 pixels wide and 8 pixels high) In a 6x8 font, characters are somewhat detailed, but you can't display a lot of text on the home screen. On the other hand, the Ti-83+ "Small Text" font has characters that average to 3 pixels wide and 5 pixels high. This means the characters are low on detail, but one can see more text at once.

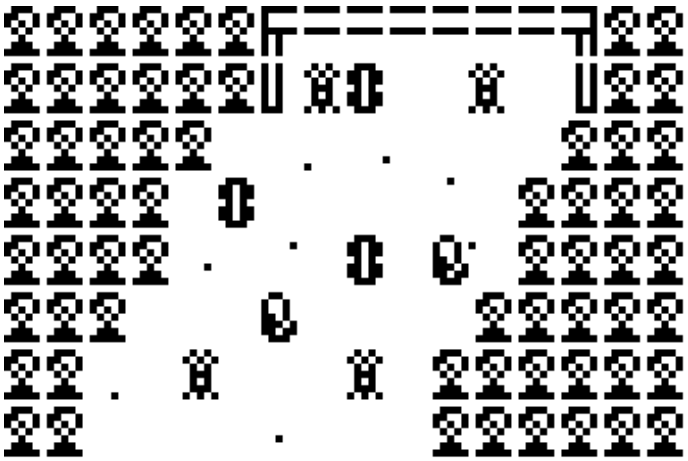
So if you want a game with detailed images, you should use a large font—16 wide by 16 high is very common for this. If you want a game that will display more, you'll want a smaller font. A lot of people use 8

wide by 8 high characters for this purpose. **I have found the 6x8 homescreen font to be a very, very common font size as well.**

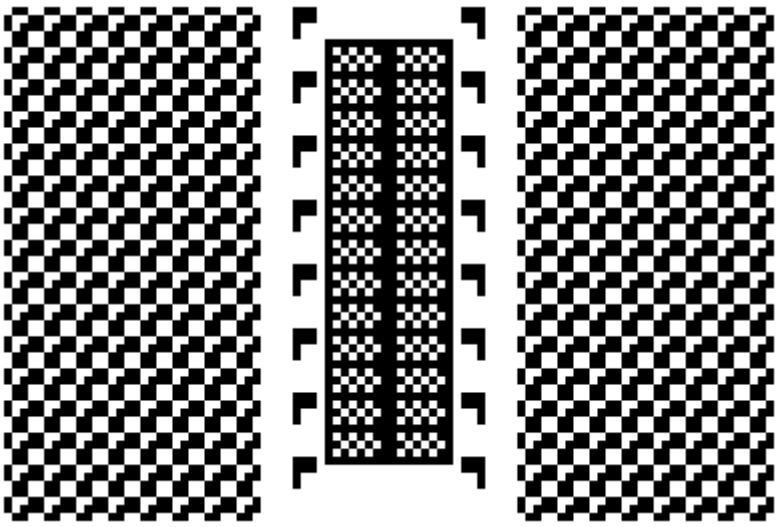
A game using 16 x 16 sized characters as a font size with no blank spaces between characters. Notice that only 24 characters—6 across and 4 down—can be displayed at once.



A game using the common 6x8 sized homescreen font size, with a horizontal blank space and a vertical blank space between each character. Notice that it can display 12 characters across and 8 characters down, although the detail is slightly less because of smaller characters. By the way, notice what horizontal and vertical blank spaces do to the checkpoint at the top of the picture—it looks like someone took a chainsaw to it and cut it into several pieces.



A 4 x 4 font with no blank spaces between characters. Although the detail is almost non-existent and the string size is large, one can display 24 characters across and 16 characters down.



All characters in your font must be the same size. Furthermore, your characters can be no bigger than 16 pixels and 16 pixels high.

Once you decide on the font size that you would like to use, you will need to decide if you want any horizontal or vertical spacing

between the characters. As a good rule of thumb, you will want to have horizontal and vertical spacing for any text-based characters you place inside of your font, so that your text is easy to read. If there are graphical-based characters inside of your fonts, you will usually want to avoid having any spacing so that your game looks like a complete image. This is not a hard-and-fast rule, however, just a suggestion. Notice on the above image, for example, there's an intentional blank space between the flags and the road because it makes the world look nicer and clearer. Similarly, on the image of the racecar track, blank spaces help us in determining the difference between the road block sign and the car next to it.

In this sample font, all letters of the alphabet are separated by horizontal and vertical white spaces (represented as red lines in this picture). That way text can easily be readable. (DO NOT add red to your custom font, by the way, no matter how many times you see it in the examples.)

With the graphics, however, there are NO blank spaces between characters of the font. When you create a world with dirt, rivers, signs and roads, you want everything to look seamless. Remember that at the bottom of page twelve, the "checkpoint sign" on the racecar track had horizontal and vertical blank spaces, and it did not look pretty whatsoever.

So keep these ideas in mind. Once you decide on a font size (and for practice, I highly recommend 6x8 or 8x8), open up your paint program. Start by setting the width of your bitmap to the width of your choice. Your height should be the height of each character, multiplied

by the number of characters you want. Notice from the sample font that characters are placed one after the other in a straight line.

When drawing the characters inside of your font, each character requires its own "area" for lack of a better word. In a 6 x 8 font, every character requires space 6 pixels wide by 8 pixels high. You'll want to start at X = 0 and Y = 0 on the bitmap to draw your first character. Your next character will be at Y = Character Height, X = 0. Then the character afterwards will be drawn at Y = Character Height * 2, then Character Height * 3, and so on. Look at the sample 6x8 font at the top of the previous page. The character "A" is drawn at Y = 0. The character "B" is drawn at Y = 8. The character "C" is drawn at Y = 16. The car is drawn at Y = 24.

As you draw your font, remember to include blank spaces where appropriate. If your next character requires horizontal blank spaces, that character should be thinner than the width of your bitmap. Suppose you are creating a font 6 pixels wide. A character with a desired horizontal blank space of 1 should be 5 pixels wide. A character with a desired horizontal blank space of 2 should 4 pixels wide. Of course, a character 6 pixels wide on this particular bitmap will have NO horizontal blank space.

When taking into account vertical blank spaces, the same rule applies similarly. Let's say that your font consists of characters no bigger than 8 pixels high. A character of text should have at least one vertical blank space, so you would want to make the character only 7 pixels high. Then you will need to make sure there is one pixel of blank space between the bottom of the aforementioned character and the top of the next character. For graphical characters on a font with 8-pixel-high characters, you'll want to consider NO WHITE SPACES between

most—if not all—of the characters. Yes, that means placing your graphical characters right on top of each other.

IMPORTANT: Be aware of what Ti-83+ character you wish to assign the first character of your custom font to. Suppose you assign the letter "A" in the previous page's example font to the Ti-Basic word prod(. Looking at the table, ln(3,4, "prod(not(iPart(fPart(") will display the A, the B, the C, and the car. But what about the checkered flag? You can't enter a character that will display the checkered flag, so you will never be able to display the checkered flag! In this case, you'll want to force Correlation to assign the checkered flag to the next available character, the square-root sign. You can do this by making blank the character that Correlation would have assigned to the Ti-Basic character you can't type.

Assigned to prod(

Auto-Assigned to not(

Auto-assigned to iPart(

Auto-Assigned to fPart(

YOU CAN'T TYPE THIS CHARACTER IN TI-BASIC, SO IT'S NOT WORTH DRAWING A CUSTOM FONT CHARACTER AT THIS SPOT

Auto-Assigned to the Square Root Sign

Auto-Assigned to the Cube Root Sign

After you have finished drawing all the characters that you would want to display in your Ti-Basic game, there might be some unnecessary white space at the bottom, space from making the height of your bitmap too big. You'll want to clip this off, but remember to leave vertical blank space for you bottom character if it requires vertical blank space.

Make sure your font has all the characters and blank spaces that you want. After words, you'll want to adjust the width of your bitmap as follows: If your bitmap is 8 pixels wide or less, the width of your bitmap should be adjusted to be **exactly** 8 pixels. Otherwise, your bitmap width must be 16 pixels. Save your bitmap as a monochrome bitmap and DO NOT COMPRESS IT. Also, since this will go to your calculator eventually, **the first character of your filename should be a letter, and the rest should be letters or numbers**. A8SEIFILCMWOW.bmp is a valid filename, 3_SL3__.bmp is not.

That's it! Your font is done! However, you can't put a bitmap onto your calculator, so read the next chapter to learn how to compiling your font into a Ti-83+ program that CAN be placed on your calculator.

Compiling a Font

Correlation comes with a CLI font compiler and a GUI font compiler, both of which let you compile a monochrome bitmap font into a Ti-83+ program that you can use with Correlation. Both programs are available in case one works and the other doesn't. You will need java, and you will also need to make sure that your bitmap and the programs are in the same folder.

Running CorrelationFontCompiler.jar is straight forward once you know how to run CorrelationFontCompiler.class, so I will mostly provide instructions for using the CLI version of the compiler. Open a CLI, such as Windows Command Prompt, and go to the directory where the program is located. Type in java CorrelationFontCompiler.

Under "bitmap image to parse," type in the name of your bitmap, including ".bmp". Note that if your bitmap is not in the same folder as CorrelationFontCompiler.class, you will need to include the location of the file.

For "width of each character," type in how many pixels wide each character of your font is. For "height of each character," type in how many pixels high each character of your font is.

For the next step, you will see "Starting Character Number." On the table where you see all the TI-83+ font characters, choose the character that you want to assign the first character of your font to. Then use the columns and rows to find its hexadecimal value. (For example,

“H” is 48, since it’s on column 8 and row 4) Convert this number to decimal and enter this as your starting character number.

Now take this number, subtract 1, and add the number of characters in your font. This will be your “Ending Character Number.” Enter this in.

That’s it! Your font will compile to a Ti-83+ program, and the name of the program will be taken from the name of your bitmap. If your program is less than 8 characters, 0s will be added at the end of the program name, since your program name MUST be exactly eight characters long.

If you are using the GUI version of the font compiler, you must specify a name for your program. Remember that the name must be 8 characters long, and a valid program name. Sorry, you cannot use Theta in the name.

Using Your Font in a Ti-Basic Program

The important thing to remember about Correlation is you can display custom-font text in a manner very similar to displaying Ti-83+ text in a Ti-Basic program. There are some lines of code you have to add to your TI-Basic program, but other than that the transition to Correlation is very smooth. ***In addition, if there's a great Text-Based Ti-Basic game that you want to convert to use a custom font with Correlation, it is not hard to do so using the instructions on these next two pages***.

If you want to use Correlation, you require two programs: pgrmCORELATE and pgrmFONTROUT. pgrmCORELATE must be stored in RAM, and pgrmFONTROUT must be stored in the archive. (If you put it in RAM, Correlation will archive it.) Your font can be stored in either RAM or in the archive.

You must have the line 1:Asm(pgrmCORELATE before Correlation will run in your program. You don't have to start your Ti-Basic program with this line, but it's not a bad idea to do so. This line prepares the Ti-Basic program to use your custom font.

Also, you must have the line 0:Asm(pgrmCORELATE before the end of your Ti-Basic program. Thus, if there are several areas where your program could end, you must have 0:Asm(pgrmCORELATE at

each of those spots. The line closes Correlation so that your calculator can run normally again.

While you can use several fonts in your Ti-Basic program, you can display text using only one font at a time. Use the instruction real(to select your font. real(must be followed by the eight characters of your font name, NO quotation marks. For example, if your font is stored in pgrmTESTFONT, use real(TESTFONT to select that font for displaying text. You can use real(at anytime during your program to select another font.

Remember that you use ln(to display text in a similar manner that Output(does, and you use e^(to display text in a manner similar to Text(. ln(will only work with a 6x8 font, but Text(can work with any font size up to 16 x 16.

The last thing that you need to know for right now is that after you use ln(and e^(, you will not see your text immediately. Normally, when you use Output(and Text(, every line of text drawn is shown immediately, and that slows your Ti-Basic program down. Correlation takes care of this by not displaying text drawn with ln(and e^(until you ask it to. Rather, it prepares it, so that when you are done drawing all your text and use **DispTable** (maybe DispGraph instead) to display it, all text will show up at once. This will greatly increase the speed of your program.

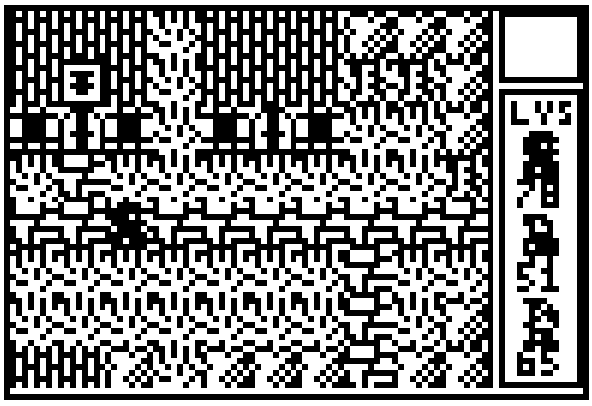
You can also display text ready to drawn by using any of the TI-Basic drawing and graphing commands, such as Pt-On(. Note that if there's any text you would not normally redraw in a Ti-Basic program, you do not need to redraw it with Correlation.

Special Ln(and e^(Techniques

*These are advanced techniques to use when you understand the basic concepts of Correlation. I do not recommend these techniques for people using Correlation for the first time. Also, I also recommend they only be used when you are making a fresh TI-Basic game—**using them on an already-made TI-Basic game is strongly discouraged.***

- You can specify a negative value for X and Y coordinates with ln(and e^(. Your negative values can be as low as -9999. This method will also display text faster than when you had to use sub(for strings displayed with the normal Ti-83+ text. Note that a value of 0 in ln—and ONLY ln—will translate to -1, and a value of -1 will translate to -2, and so on.
- Text with ln(and e^(can be displayed in four ways: Clip, Wrap, Word Wrap, and Map.
    - In clip mode, text will stop displaying once the edge of the screen is reached. If you ever used Text(in a Ti-Basic program, that normally uses clip mode.
    - If text is displayed using Wrap, characters will be drawn until a line is full, after which the characters will continue onto the next line. If you ever used Output(in a Ti-Basic program, it normally uses Wrap mode. This is also the default mode used for ln(and e^(.

- Word Wrap is a special mode that takes a string and draws it Word Wrap style—that is, moving text from line to line to create an easy-to-read paragraph. The game will then pause and allow the user the scroll text up and down until enter is pressed. This allows a user to create storyline with long text without having to make a bunch of complex calculations. The catch is that you must have ASCII character 32 in your font to allow spaces. That means your starting character value must be 32 or less, and your ending character value must be 32 or more. You also cannot use negative values for X and Y in word-wrap mode. X cannot be greater than 8 for ln, and (96 – (Character Width * 8)) for e^. Finally, strings displayed using Word Wrap cannot take up more than 50 lines.
- Map mode is another special mode that is meant for games such as Text-Based RPGs. Normally, you probably used multiple strings and/or sub(to display a text-based map that you could move around. However, with Map mode, you only need one string (containing the text you would use to display a map) and a map-width, such as “30 characters wide.” Correlation handles the rest so that your map comes out right without any weird text wrapping or clipping. If you wish, you can also have your map drawn on only part of the screen.



Here, the map doesn’t draw over the entire screen in this screenshot. That way, the status bar on the far right won’t be erased.

- To switch to Clip Mode in Ti-Basic programs, use GridOn. To switch to Wrap Mode, use AxesOn. Use LabelOn to switch to Word Wrap, and use ExprOn to use Map Mode. Note that in Map mode, you will need Xscl to specify a map width, which cannot be bigger than 255. Also, use Xmin, Xmax, Ymin and Ymax to specify where on the screen you want drawing of your map to start and end. Whether you are using ln(or e^(, the values for min and max are in PIXELS. Also, for Xmax and Ymax, the value is the first pixel where drawing for a row or column STOPS. Look at the above screenshot, where the map doesn't cover the entire screen. In this case, Xmin = 0, Xmax = 80, Ymin = 0 and Ymax = 64.
- When you are using Wrap Mode, a negative Y value will cause off-screen text to wrap as if the text were on screen.

Advanced Text Graphics

*These are advanced techniques to use when you understand the basic concepts of Correlation. I do not recommend these techniques for people using Correlation for the first time. Also, I also recommend they only be used when you are making a fresh TI-Basic game—**using them on an already-made TI-Basic game is strongly discouraged.***

By default, any text that you draw will erase anything underneath it. You have probably seen this when you used Output(and Text(in your old Ti-Basic programs. This method of displaying text is called OVERWRITE, because the text erases everything underneath is.

When you draw text from a font with graphical characters, however, you might want to have some transparency so that you don't erase the background. Correlation allows you to draw text with transparency whenever you use ln(and e^(. All you need to do is add a fourth parameter.

Suppose you want black pixels in your font to be drawn and white pixels to be transparent. (This is called OR.) Just add a 4th parameter of 1.

ln(2,7, "EISFE",1)

e^(45,12, Str0, 1)

If you add a 4th parameter of 2, your font will be drawn with an XOR method. White pixels will be transparent, and black pixels will INVERT the colors of anything underneath them.

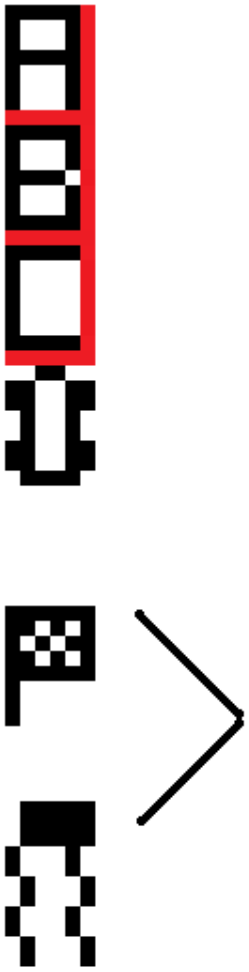
If you add a 4th parameter of 3, your font will be drawn with an AND method. White pixels will be drawn normally, and it's the black pixels that will be transparent.

To switch back to OVERWRITE mode, use a 4th parameter of 0.

Sometimes you'll have a graphical character in your font where you want some parts white, some parts black, and some parts transparent. If you want to do just that, you need a 4th parameter of 4.

e^(32,57, "ACEB", 4)

This method is called drawing a MASKED font character, and is usually used for drawing a graphical image where you want black, white and transparency all at once. However, if there's a graphical character you want to draw MASKED, you have to make some adjustments to your font. A Masked font character requires a regular character and a mask to go with it. So two character's worth of bitmap font space is required—the first space for your regular character, the other space for its mask.

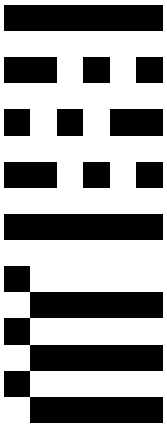


We want the checkered flage to be black, white and transparent. So we need TWO characters for it.

For the first part of your masked font character, any part that you want white or transparent should be colored white, and any part that you want black should be colored black. For the second part of your character, paint any transparent part black and any other part white. Notice in the example above that in the "mask" part of the checkered flag in the font, the transparent area is black and everything else is white.

Now comes the lengthy part. You need to shuffle some lines in your font. Start by making your bitmap 8 pixels or 16 pixels according to the instructions on page ___ in "Designing a Font." Then, starting at the row where your character begins, you need 8 pixels of "regular character" data followed by 8 pixels of masked data followed by 8 pixels

of regular character data and so on and so forth. Note that this “magic number 8” applies **even to fonts larger than 8 pixels.** If you do everything correctly, your character and it’s mask should still take no more than 2 character’s worth of space.



This is the finished product of taking your checkered flag and mixing it with its mask.

Since drawing mask-able font characters is a long process, don’t abuse them. Believe me: you don’t need a lot of masked sprites for a game.

Tips, Tricks and Optimizations

*These are advanced techniques to use when you understand the basic concepts of Correlation. I do not recommend these techniques for people using Correlation for the first time. Also, I also recommend they only be used when you are making a fresh TI-Basic game—**using them on an already-made TI-Basic game is strongly discouraged.***

- There are only so many characters available on the Ti-83+ keypad. With the fact that you might be forced to click menu after menu to retrieve values such as product(and solve(for some font characters, a _____(Number of Characters coming soon)____ font can be hard to work with. A huge font is indeed necessary for some games, but whenever you can, try to have several fonts using characters A-Z (and other easy-to-reach characters) as opposed to one big font.
- Real(is the only Ti-Basic command that has a strict format in Correlation. Other than that, if there are optimizations you like to use in Ti-Basic programs, you can use those same optimizations with Correlation commands.
- The only time that you require a 4^{th} parameter for Ln(and e^(is when you need to change the way your font is displayed. If your plan is to display fonts using the same display method over and

over, you only need specify it with a 4th parameter once, after which Ln(and e^(will default to it. This will save program space and help your program run faster. The following is unnecessary:

Ln(1, 1, Str0, 3)

Ln(2, 2, Str1, 3)

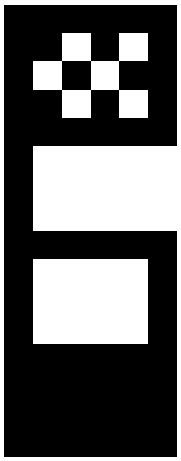
Ln(3, 3, Str2, 3)

Rather, you need to specify "3" as a 4th parameter only on the first line, and all three lines will draw with AND. You also do not need a 4th parameter when switching between Ln(and e^(, as long as you plan to use the same display method for both routines. Just be sure to include a 4th parameter to change to a different display routine. Remember that when your TI-Basic program starts, the default display routine is OVERWRITE.

- If you have a library such as Xlib or Celtic III that you want to use for your Ti-Basic program, you can use it in conjunction with Correlation. However, you must temporarily switch from Correlation to your other library. Use CoordOff to switch from Correlation to another library, and use CoordOn to move to Correlation again. Be aware that Correlation does not allow you to use, for example, Xlib and Celtic III with each other.
- Correlation uses what's called a **parser hook**. Parser hooks always slow down Ti-Basic programs, even if slightly. If you want to get the greatest amount of speed from your Ti-Basic game, consider turning Correlation off when you don't need to use Ln(or e^(for text for more than 5 lines. You can turn Correlation back on when you need to. To turn Correlation off, use ZoomRcl, and

use ZoomSto to turn it back on. Be aware that ZoomRcl will turn off Xlib, Celtic III, etc. as well.

- If you are creating a mask-able character, you may occasionally come across a situation where you want parts of your masked character to **invert** pixels underneath when the character is drawn. To do this, paint black this area in BOTH your normal character and its mask.



The outline of the flag, as well as the flag pole, will now invert any pixels that are underneath them when the checkered flag is drawn.

Error Messages

Occasionally, you may come across an error message that you've never seen in Ti-Basic before. These error messages are provided by Correlation to help you find errors that you may have made. This section of the guide will explain error messages and the mistakes that you should fix inside of your program as a result.

BASIC ERROR MESSAGES

ERR: NO FONTROUT

You will receive this error if your Ti-83+ is missing pgrmFONTROUT. To use Correlation, pgrmFONTROUT must be located on your calculator.

ERR:NO SUCH FONT

You are attempting to use real(to load a font that doesn't exist. Make sure that you typed the font name correctly and that the font is indeed on your calculator.

ERR:OUT OF RANGE

Every font has a range of characters you can display using strings. If you attempt to go outside of this range, you will receive this error. Correlation will return an error, for example, if you try to use an "!" for a font that only allows A-Z.

ERR: WRONG SIZE

When you use Ln(to display strings, the characters inside of your font must be 6 pixels wide by 8 pixels high. You must select another font to use if the characters do not follow this format, or use e^(instead.

WORD WRAP AND MAP MODE ERROR MESSAGES (FOR EXPERTS)

ERR: WORD WRAP

You are using X and Y coordinate that are not valid in Word Wrap Mode. Make sure that you really wanted Word-Wrapping rather than a different mode.

ERR: TOO LONG

Word-wrapping only works with text that takes up to 50 lines. You will need to shorten your text—or display it further to the left—if it takes more than 50 lines. Lines are counted as you scroll the text down, so make sure you test for an error by scrolling to the bottom of the string

that you displayed. You only need to check for this error once per string that uses Word Wrapping.

ERR: NO SPACE

When you use Word Wrap mode, your selected font must have a starting character value 32 or less, and an ending character value 32 or more.

ERR: BAD WINDOW

In map mode, Xmin must be less than Xmax, and Ymin must be less than Ymax. Also, NONE of these values can be negative. Finally, Xmax cannot be bigger than 96, and Ymin cannot be bigger than 64.

Special Thanks and List of Testers

SPECIAL THANKS TO THESE PEOPLE FOR MANY DIFFERENT THINGS, BUT ESPECIALLY THE FOLLOWING (IN NO PARTICULAR ORDER):

thepenguin77—General parser help, converting BCD to decimal

calc84maniac—Converting BCD to decimal

Sean McLaughlin—Sprite Routines

Iambian, calcdude84se—Guidance on drawing overwritten sprites

Xeda112358, BrandonW, KermM—General parser help

DJ Omnimaga—Countless suggestions and hints that helped make this a high quality package

Graphmastur—Special hook help

BuckeyeDude—VAT and variable help, general parser help

Deep Thought—Main Programming of the Correlation Font Compiler

Nemo—GUI of the Correlation Font Compiler

Romain Liévin, Tim Singer—Ti-83+ Character Table

AND SPECIAL THANKS TO THE TESTERS:

COMING SOON!

Screenshot Credits

COMING SOON!