Gif description V01

1 Introduction

This document describes how the GIF VI's work. With version 01.00 it's only possible to read a GIF file. The detailed GIF format description can be found in [1]. All information to create this GIF functions are from this document.

2 Overview

List of all used VI's in version 01.00.

Gif.Example.vi
Gif.Main.vi
Gif.Header.vi
Gif.LogicalScreenDescriptor.vi
Gif.ImageDescriptor.vi
Gif.DataSubBlocks.vi
Gif.ColorTable.vi
Gif.GlobalColorTable.vi
Gif.LocalColorTable.vi
Gif.getBlock.vi
Gif.GraphicControlExtension.vi
Gif.CommentExtension.vi
Gif.ApplicationExtension.vi
Gif.ImageData.vi
Gif.CreatePictureArray.vi
Gif.CreatePicture.vi
Gif.ExtractTime.vi
LZW.decode.vi

3 Gif Layout

The Gif layout describes where which part of the Gif file can be read. For detailed information about it see [1] Appendix B.

4 Detailed Description

This chapter describes all VI's in detail, included are connector pane, block diagram, front panel and detailed description of the functionality.

4.1 GIF.Example.vi

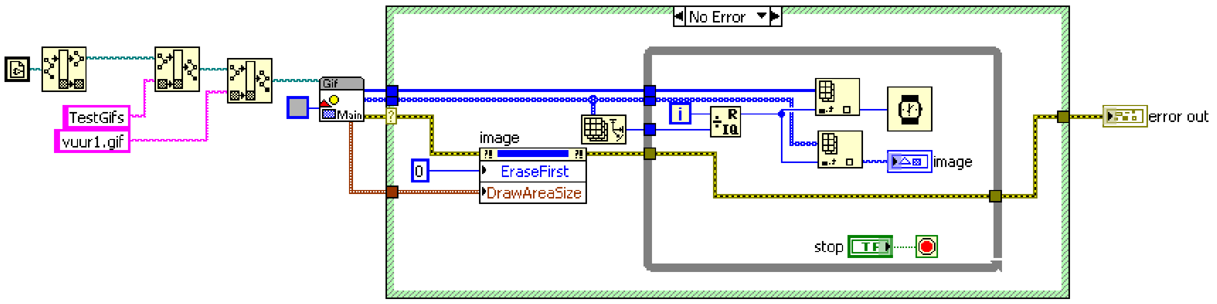

Figure 1: Gif.Example.vi Blockdiagram

This VI calls a specified Gif file, run the Gif.Main.vi and shows the single pictures in a LabVIEW image. The Wait time in the loop is the time from the Gif. The loop can be stopped manually. To avoid flickering the "erase first" property is set to "0". More about LabVIEW images can be found in [2].

4.2 Gif.Main.vi

The Gif.Main.vi is designed as a state machine. The first state is "readfile". Figure 2 shows that the specified file will be read as a binary file. The result is stored in a shift register to use it in other states.

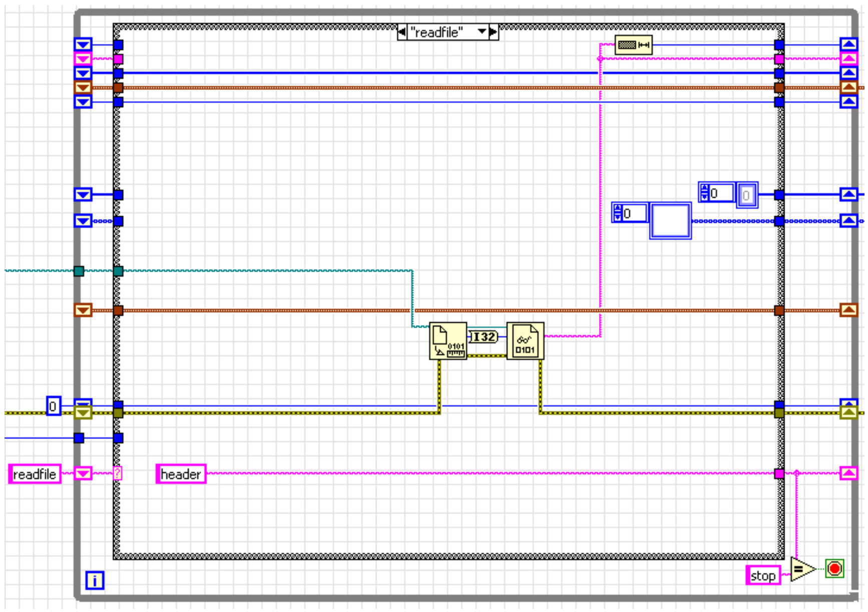

Figure 2: Gif.Main.vi - "readfile"

Shift register description from top to bottom:

- File size: we don´t need it!

- File data: that´s where we get all information from
- Global color table: contains the colors from the global color table. Can also have size 0!
- Logical screen descriptor: contains information about size, global color table and other properties of the image. For detailed information about it, see [1]!

- Wait time: contains the delay time of all images
- Image: contains the entire images

- Graphic control description: contains the delay time of the single image, has to be before every single element in the gif file

- Position: contains the current position from which the next functions reads the data from the file data
- Error cluster

- State: to switch between the different states (needed for a state machine, can also be an enumeration!☺)

The next state is state “header”. In this case the first six bytes of the file stream will be read and evaluated. If it´s not a gif file or the version of it is other than 89a, then one of the error states will be called. There are two error cases, one for a version error and one for a wrong file format error.

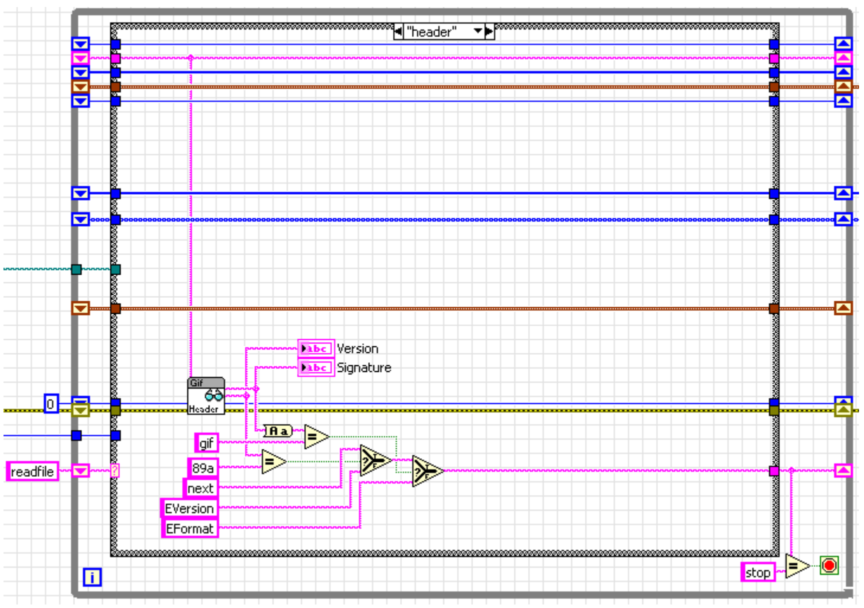

Figure 3: Gif.Main.vi - "header"

If no error occurs, then the next case is “next”. In this case the logical screen descriptor will be read. If a global color table exists, then it will follow directly after the logical screen descriptor. The last sub VI call in this case, is to initialize the Gif.CreatePictureArray.vi. The information from the logical screen descriptor and the global color table will be stored in shift registers. The background color, specified by the user, will also be stored in a shift register.

This shift register can later be moved inside the action engine (Gif.CreatePicutreArray.vi).

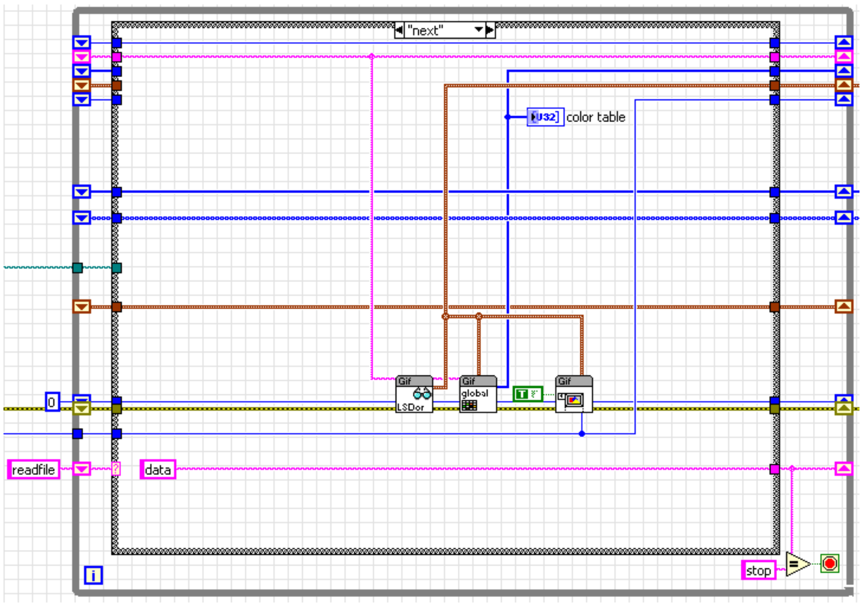

Figure 4: Gif.Main.vi - "next"

The next case is the "data" case. The "data" case will be called until the "exit" command will be read or an error (found no specified marker) occurred. Inside this case is another case structure. One case is the "extension introducer (value = 33 or x21). The extension introducer will be available if an extension block follows. Other cases of the inside case structure handles the errors, the end marker or the image. The image descriptor has the marker x44. Inside this case the real image will be read.
The first sub VI is to read the image descriptor. After this, if available the local color table will be read, followed by the VI which read the real image data. All these information are then inputs of the Gif.CreatePictureArray.vi, where the entire image array will be build. The image array will be transformed into a LabVIEW image and the delay time will be extracted. The image and the delay time will be added to an array.
If all images are read, the function will find the termination character. The last case is the exit case which tries to call the stop case.
If the state "stop" is selected, the while loop will exit.

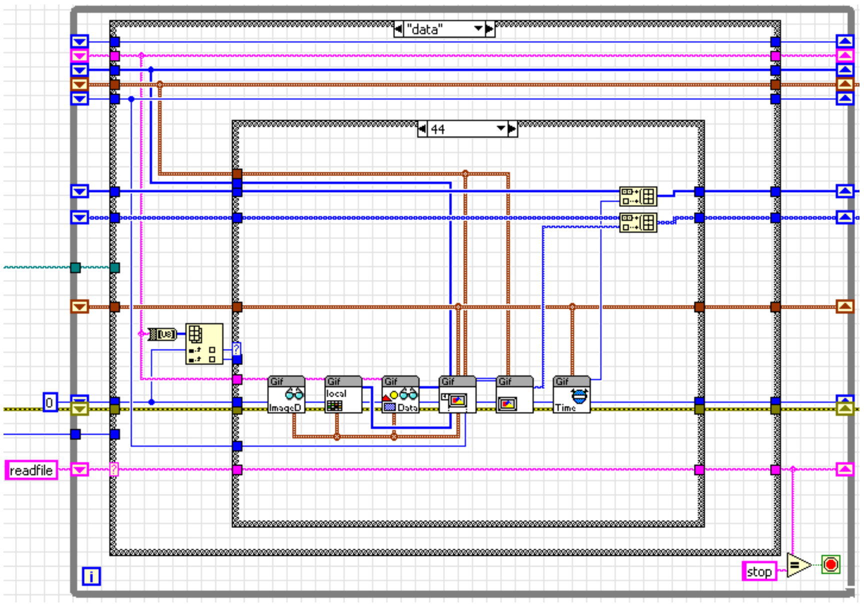

Figure 5: Gif.Main.vi - "data"

4.3 Gif.Header.vi

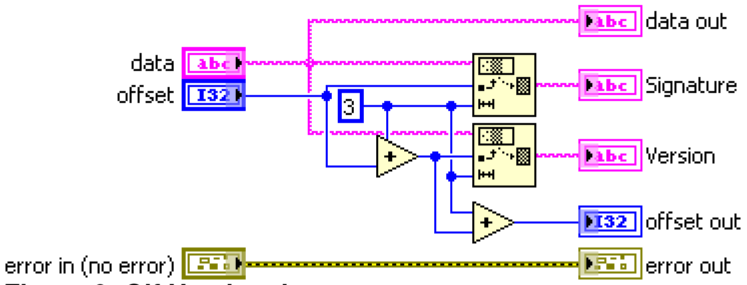

Figure 6: Gif.Header.vi

This function reads the Header information of the opened file. In a GIF file the first three bytes has to be “GIF”. The next three bytes contain the version of the file. The version could be “87a” and “89a”.

The present described VI´s can only work with version 89a!

4.4 Gif.LogicalScreenDescriptor.vi

The Logical Screen Descriptor contains properties of the resulted images, like the maximum width and height. It has to follow directly after the header information and has to be in every gif file. If the global color table flag is true, then the global color table will follow directly after the logical screen descriptor.

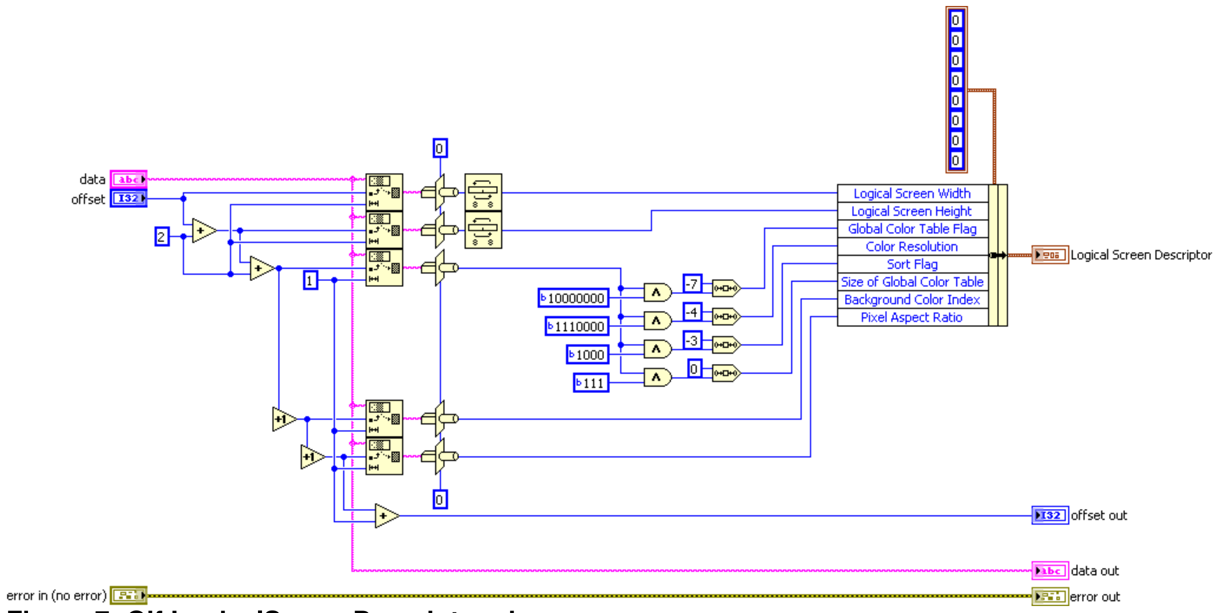

Figure 7: Gif.LogicalScreenDescriptor.vi

4.5 Gif.ImageDescriptor.vi

The image descriptor contains information about the single pictures in the gif file. Those images can be smaller than the entire result image and contain in most cases only the differences from the before shown image to the new one. Therefore it contains the image width and height as well as the image left and top position in the original image.

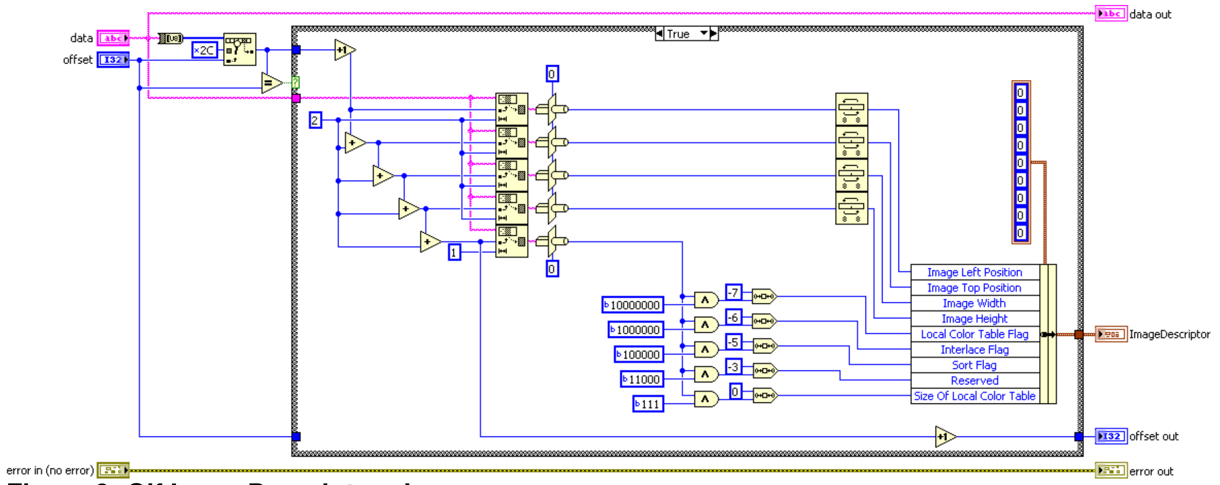

Figure 8: Gif.ImageDescriptor.vi

An image descriptor has to be directly before the image data. If the image has a local color table, then this table is located between image descriptor and image data!

4.6 Gif.DataSubBlocks.vi

This VI reads the data sub blocks. Each data sub block ends with an x00. If the end sign is found, the loop stops execution and the array “SubBlock values” contains all found data.

The callers of this VI are:
- Gif.ImageData.vi
- Gif.CommentExtension.vi

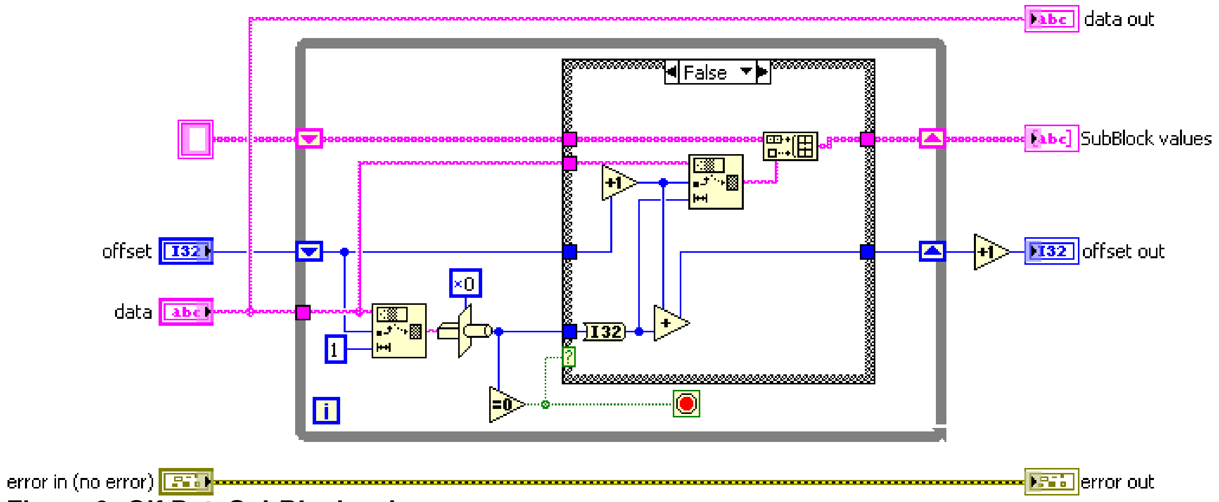

Figure 9: Gif.DataSubBlocks.vi

4.7 Gif.ColorTable.vi

Depending from the caller, Gif.GlobalColorTable.vi or Gif.LocalColorTable.vi, this vi reads the local or the global color table.

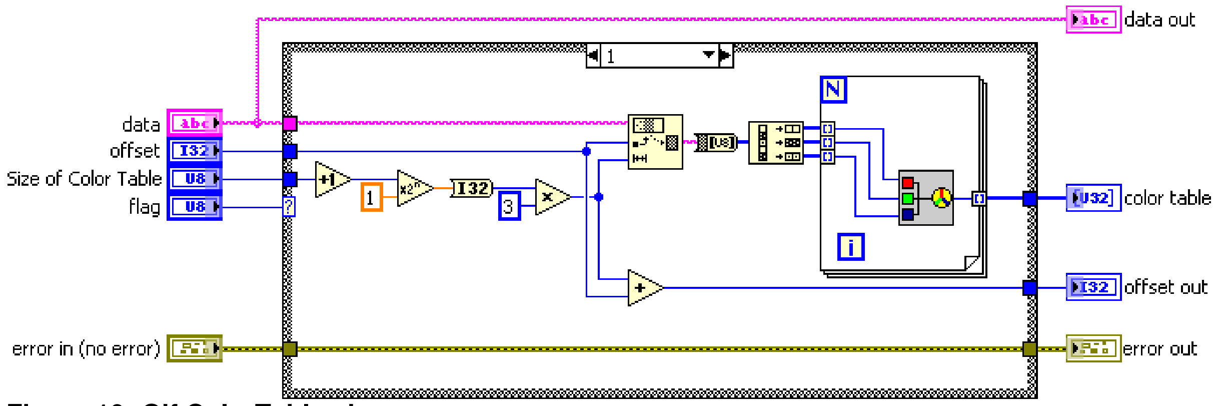

Figure 10: Gif.ColorTable.vi

4.8 Gif.GlobalColorTable.vi

This VI calls the Gif.ColorTable.vi to read the global color table from the gif file.

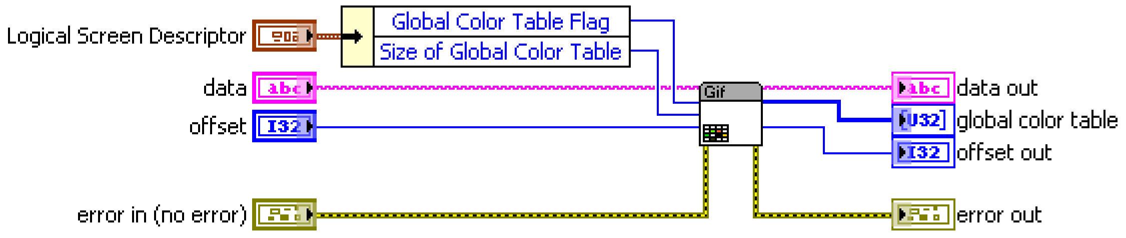

Figure 11: Gif.GlobalColorTable.vi

4.9 Gif.LocalColorTable.vi

This VI calls the Gif.ColorTable.vi to read the local color table for the current image.

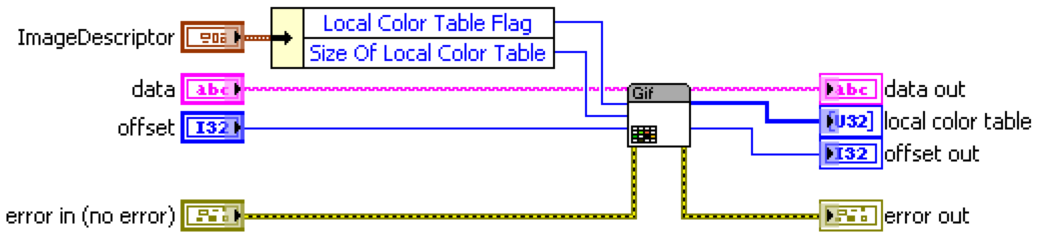

Figure 12: Gif.LocalColorTable.vi

4.10 Gif.getBlock.vi

The getBlock function reads the start position of the blocks. Blocks are defined by one of the following marks:

- x01: Plain Text Extension (**not jet used!**)
- xF9: Graphic Control Extension
- xFE: Comment Extension
- xFF: Application Extension

Callers of this VI are:

- Gif.ApplicationExtension.vi
- Gif. CommentExtenion.vi
- Gif.ControlExtension.vi.

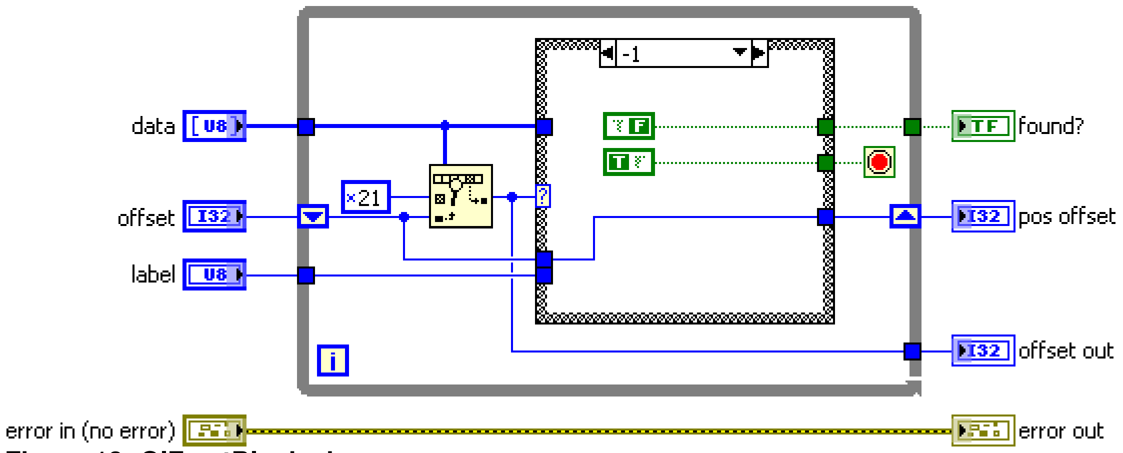

Figure 13: GIF.getBlock.vi

4.11 Gif.GraphicControlExtension.vi

The Graphic Control Extension contains data like “Disposal Method”, “Delay time” and the transparent color if used.

This information should present before each available image!

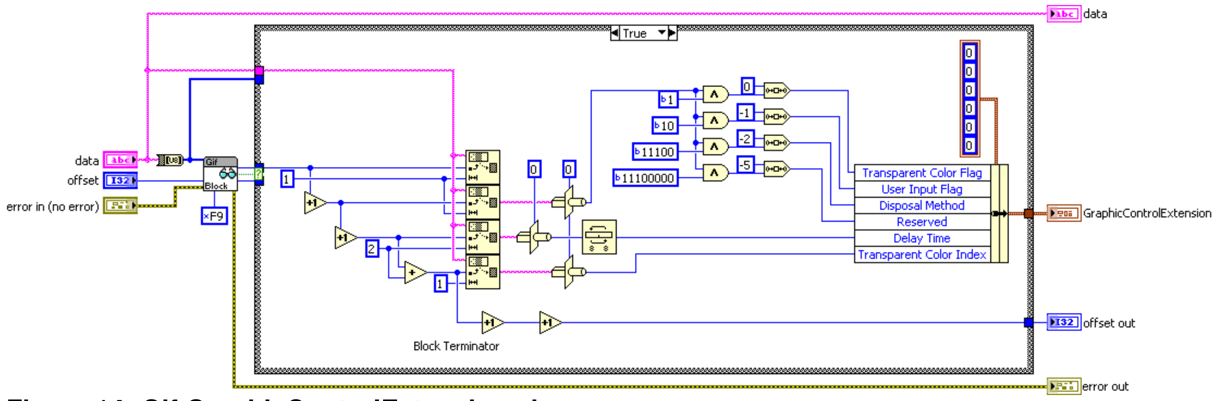

Figure 14: Gif.GraphicControlExtension.vi

4.12 Gif.CommentExtension.vi

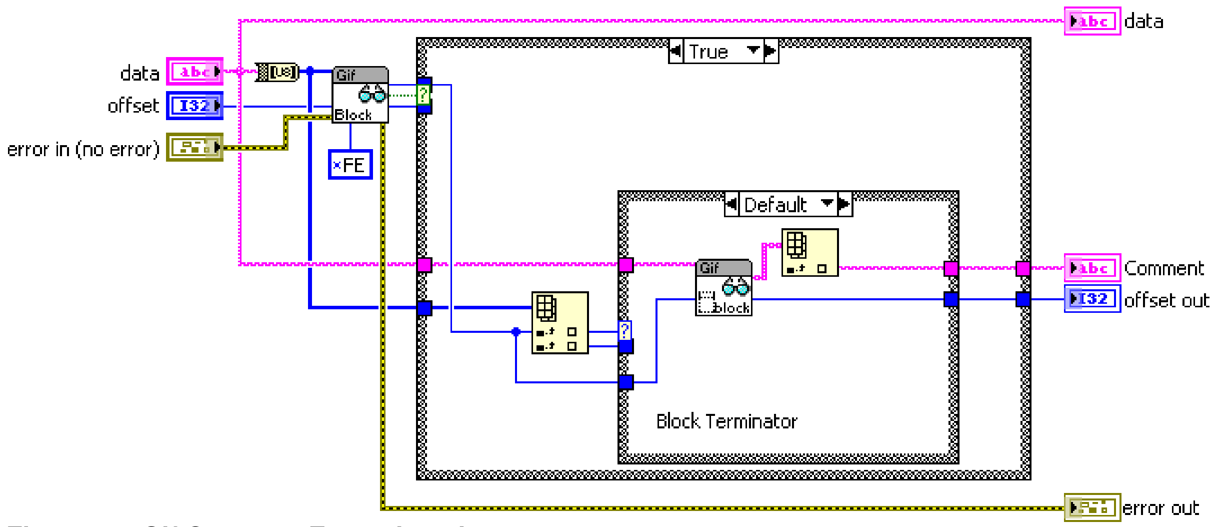

Figure 15: Gif.CommentExtension.vi

This function read the comments from the gif file. Comments can be part of the image, but they don´t have to. They are not necessary to “play” the gif image.

This function is used to get the correct offset for following blocks!

4.13 Gif.ApplicationExtension.vi

This function reads the Application Identifier and the Application Authentication Code. The data which can be available will currently not be read, but if some data is available, the offset will be set to the position after this data.

This function is used to get the correct offset for following blocks!

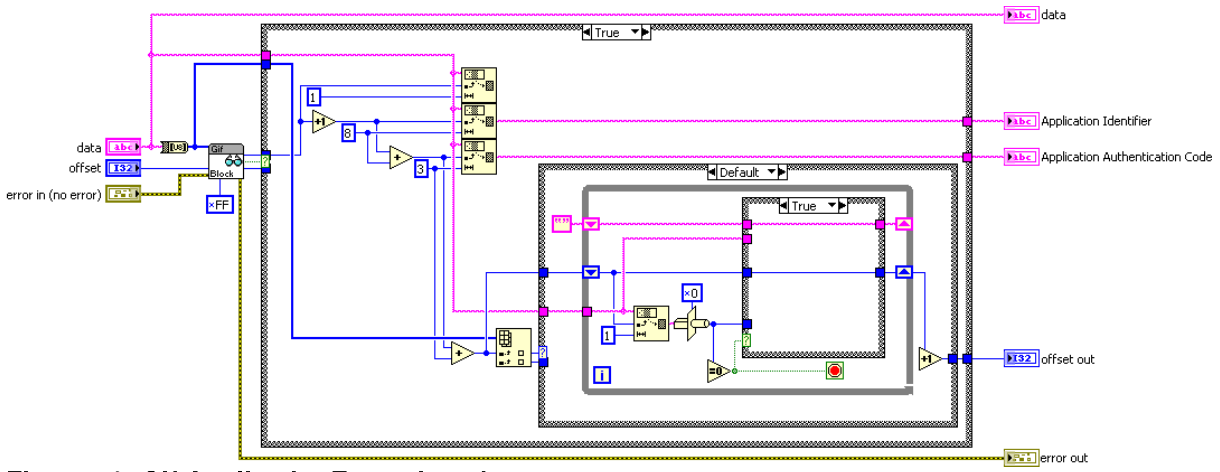

Figure 16: Gif.ApplicationExtension.vi

4.14 Gif.ImageData.vi

This function reads the image data. First it calls the Gif.DataSubBlocks.vi to get the raw file data. The resulted array will then be transformed to a binary string. This string is used to extract the image out of the raw data. The data is packed with a modified LZW code. The first byte of this data block contains the number of start bits for the LZW code. The part image size, information from the image descriptor, is used to initialize the image array.

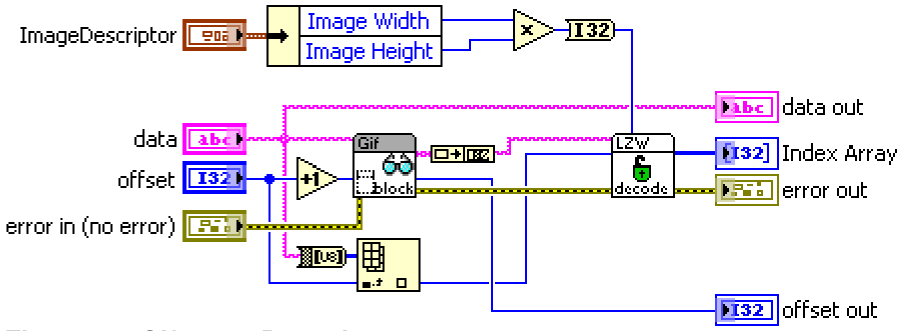

Figure 17: Gif.ImageData.vi

4.15 Gif.CreatePictureArray.vi

This function is used as an action engine. It will be initialized in the “next” case in the Gif.Main.vi. In this phase a 2D array with the resulted image size will be created. This created array is the basis for all other images.

This function creates the result image which will be displayed. The result depends on the specified disposal method, the used color table and the transparent color. If the local color table flag is true, then the local color table will be used, otherwise the global color table.

Detailed information about the disposal method are available in [1]. ☺

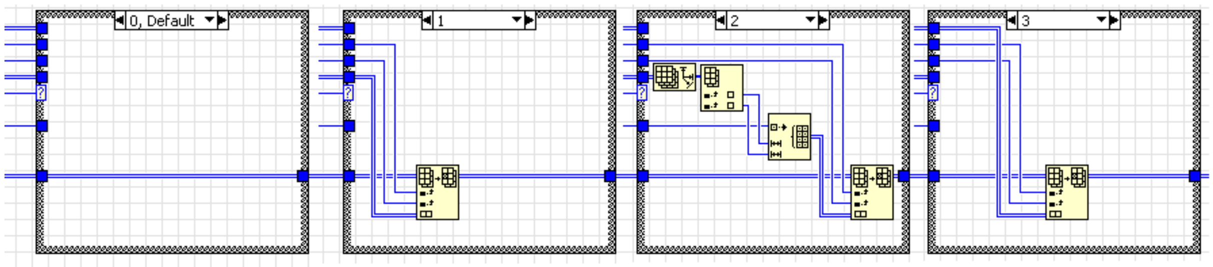

Figure 18: Disposal Methods

Case 0 – Case 3 is used to realize the different disposal methods.

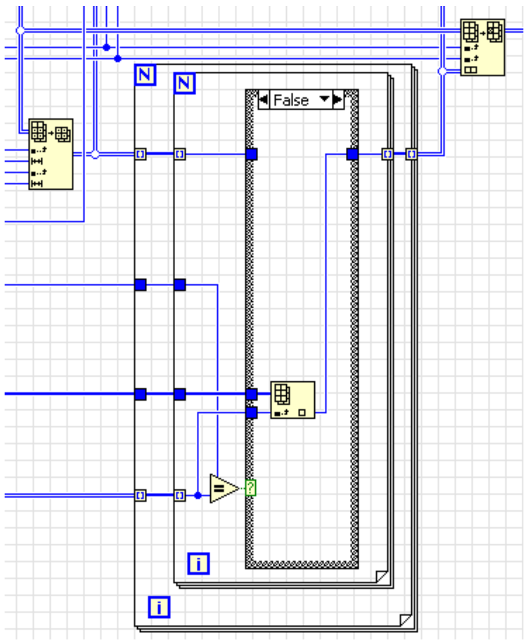

Figure 19: create new image

The array subset functions returns the part of the image which will be changed by the current read image. The two for-loops write the current color. If the current color is equal to the transparent color, then the last color is used, otherwise the new color is used. After the for-loops the current array will be inserted into the entire image array.

4.16 Gif.CreatePicture.vi

The create picture function creates a LabVIEW 24 bit image. The input “image array” contains the entire image. The 2D color image will be reshaped to a 1D array and splitted into RGB array.

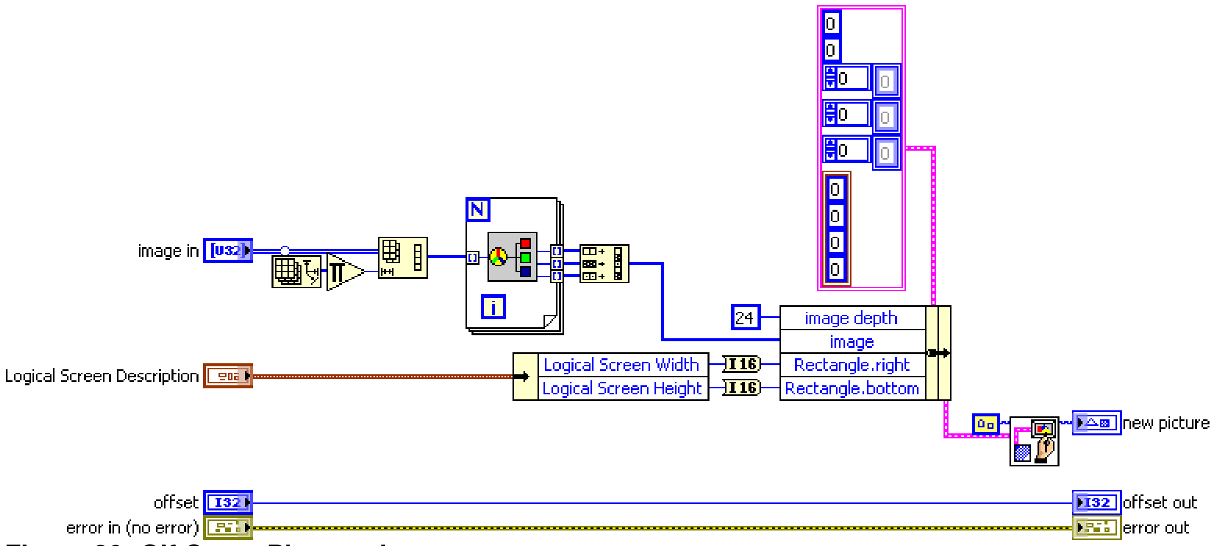

Figure 20: Gif.CreatePicture.vi

4.17 Gif.ExtractTime.vi

This function reads the delay time from the graphic control extension information and calculates the new delay time from it. If the delay time from the image is 0, then it will be set to 100, otherwise it´s multiplied with 10.

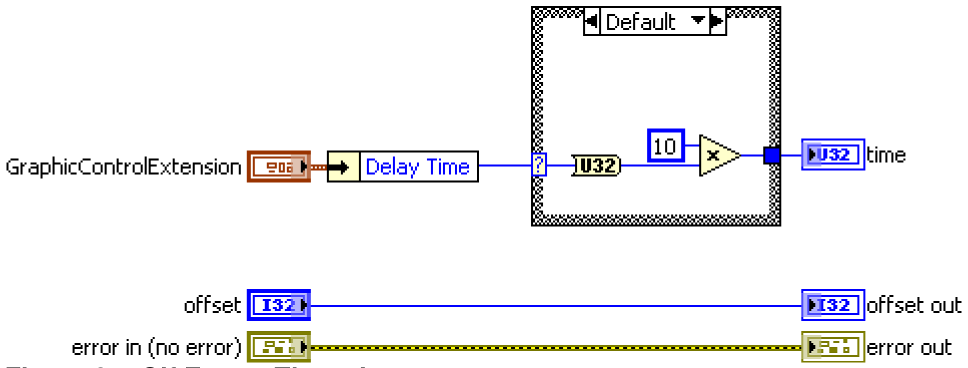

Figure 21: Gif.ExtractTime.vi

4.18 LZW.decode.vi

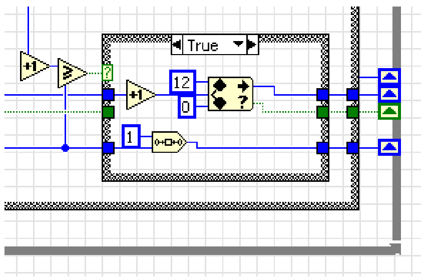

Figure 22: LZW.decode.vi – “numberofBits”

This function decodes the image data. Therefore a special LZW code is used. As described in [1] a clear code and an end of image code is used. The clear code is available in the first “number of start bits” + 1. The end of image code is one more than the clear code. The clear code resets all shift register to default and can be appear everywhere in the image data. The maximum number of bits used for one color index is 12 and the maximum dictionary size is 4096. Figure 22 shows the part of code which checks the current dictionary size. If this size is bigger than the maximum possible number of currently used bits, then this number will be increment by one.

For more information about the LZW algorithm see [3] and for additional information how to use it in gif files, see [1].

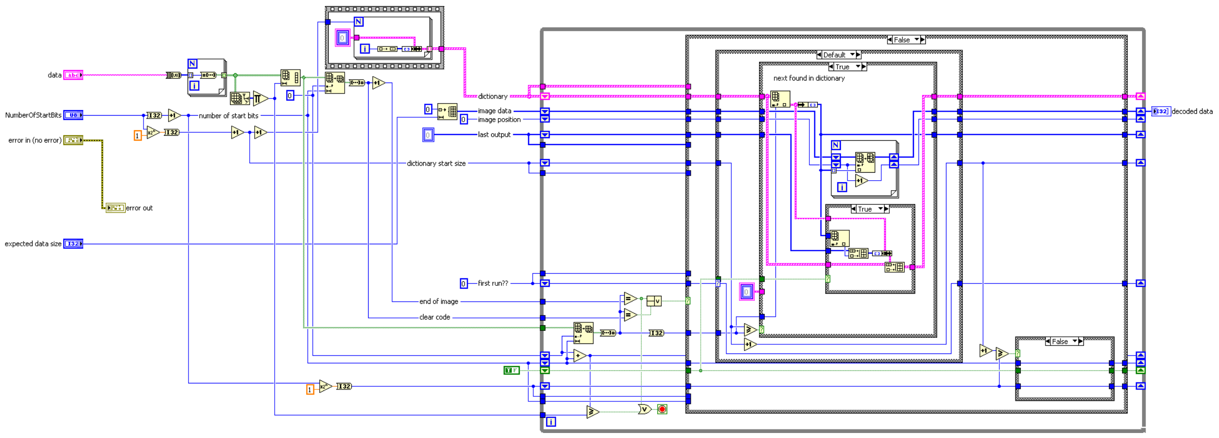

Figure 23: LZW.decode.vi

5 Table of Figures

6 References

[1] http://www.w3.org/Graphics/GIF/spec-gif89a.txt
[2] http://forums.ni.com/ni/board/message?board.id=170&message.id=212920
[3] http://en.wikipedia.org/wiki/LZW