

Using the Run Length Encoding Features on the MPC5645S

by: **Josep Martinez**
Guadalajara
Mexico

Contents

1 Introduction

Run length encoding (RLE) is a method that allows data compression for information in which symbols are repeated constantly. The method is based on the fact that the repeated symbol can be substituted by a number indicating how many times the symbol is repeated and the symbol itself. Data which is compressed using RLE method falls into two categories:

- Repeated symbols: This subset of data consists of the symbols that can be compressed by replacing them with a number indicating how many times the symbol is repeated and the repeated symbol.
- Non-repeated symbols: This subset of data consists of the symbols that cannot be compressed because they are not repeated and cannot be predicted. This is the non-compressed data.

For example, consider the following text string:

YYYYYYNNNNNNYYYYYYNNNNNNYNYN. The string consists of 30 elements with two different symbols. By applying RLE we can get the following reduced string: 6Y5N9Y6NYYN, consisting only in 12 elements achieving a compression of 2.5 times.

There are several ways to encode raw data using RLE. In the case of the MPC5645S, the data is encoded pixel by pixel, where pixel = symbol, from the starting address of the bitmap raw data as a single dimension array. [Figure 1](#) shows how this is done.

1	Introduction.....	1
2	RLE on the MPC5645S.....	2
2.1	DCU3 and DCULite embedded RLE.....	2
2.2	Standalone RLE decoder.....	3
2.2.1	Decoding non-graphical RLE data.....	4
3	Use case examples.....	5
4	Example code.....	6

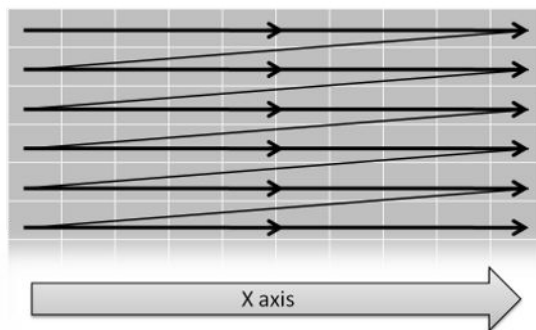


Figure 1. RLE encoding direction

2 RLE on the MPC5645S

The MPC5645S has runtime RLE decompression optimized for pixel data. The module can decode the following different pixel formats for source data: 8 bpp, 16 bpp, 24 bpp, and 32 bpp. This allows the selection of atomicity of source data for more efficient compression.

The RLE capabilities of the MPC5645S can be divided into two sections:

- DCU3 and DCULite embedded RLE: The DCU3 and the DCULite can automatically read RLE compressed data from images and display them seamlessly. Both DCU3 and DCULite have a dedicated decompression module and can be used simultaneously with the standalone RLE module.
- Standalone RLE decoder: This module allows the decompression of RLE data independently from the DCU3 or the DCULite. It can be used for any type of RLE data, including non-graphics data.

There are different ways to compress data using the RLE method. The MPC5645S RLE compressed data consists of frames of two elements: a command byte and the data.

- The command is an 8-bit value: CMD[7:0].
- The most significant bit, CMD[7], indicates if the following data is compressed or not.
- The remaining bits, CMD[6:0], indicate the number of pixels that will be decoded. This number is offset by 1 so the number of pixels, $N = \text{CMD}[6:0] + 1$.
- The data atomicity will depend on the selected pixel format, 8, 16, 24, or 32 bpp.
- The data size when it is compressed, $\text{CMD}[7] = 1$, will be always one pixel. This means that $\text{CMD}[6:0] + 1$ pixels need to be unrolled.
- The data size when it is not compressed, $\text{CMD}[7] = 0$ will be always $\text{CMD}[6:0] + 1$.

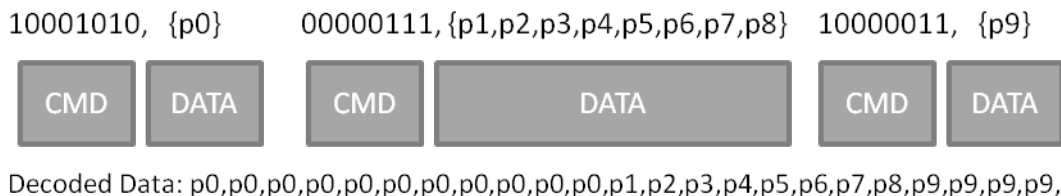


Figure 2. RLE compressed data example

2.1 DCU3 and DCULite embedded RLE

The DCU3 and the DCULite have the ability to decode and display RLE compressed images on the fly. Using an RLE image has the following limitations:

- Can be used only on either layer 0 or layer 1 on a single layer
- Supports 8 bpp, 16 bpp –except YUV, 24 bpp, and 32 bpp
- Tile mode is not permitted when RLE mode is active

From the application point of view, there is no difference between using a non-RLE and an RLE image except that by using an RLE image a lot of memory space can be saved. It can be very useful if a video or an animation needs to be played and all of the images can be compressed, or the image dimensions are big and it has a good compression ratio.

Configuring the DCU3 or the DCULite to display an RLE image is similar to configuring it to display any other kind of image. You must only perform the following additional steps to the layer initialization:

- Set `DDR_MODE` in `DCU_MODE`
- Set the compressed size of the bitmap on the Compression Image Size Register `COMP_IMSIZE`
- Set the `RLE_EN` bit on the control descriptor 4 register `CTRLDESCLn_4`

2.2 Standalone RLE decoder

RLE compression is not only useful to compress images but other types of data such as sound and program code. The standalone RLE decoder in the MPC5645S allows decoding any type of data whether it is an image or not.

To decode RLE data, configure the eDMA channels to move the data to and from the RLE decoder module. DMA transfers will be triggered when required by input and output FIFOs.

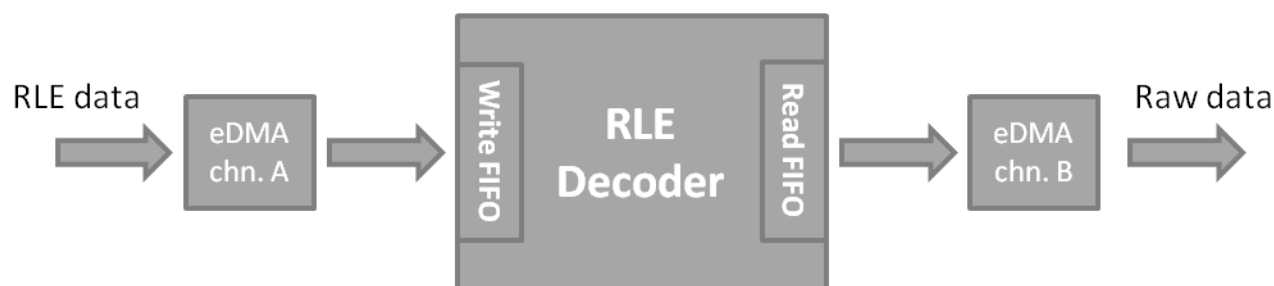


Figure 3. RLE application block

The RLE data in application diagram shown in [Figure 3](#) can be fetched from any of the available memories and the decoded data can be put into SRAM, GRAM, or DRAM memory.

When images are decoded, the RLE Decoder also has the ability to extract a smaller image or portion of the original RLE-compressed image. The images are divided into a virtual grid of pixels where it is possible to select the start position and end position. [Figure 4](#) is an example of an image with the grid division.

NOTE

The grid start point is located at (1,1), the end point is located at (width, height). The start and the end points of the selection rectangle are included in the partial image area.

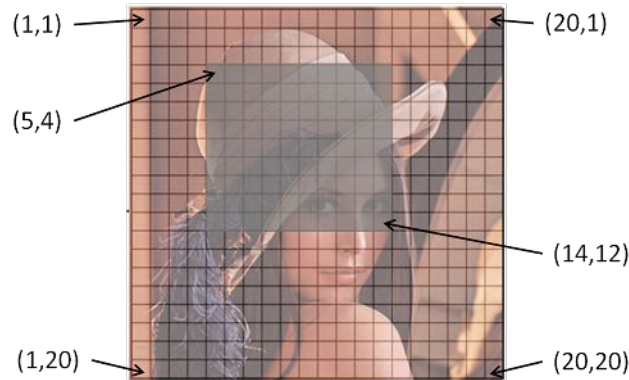


Figure 4. 20x20 pixels image example with grid

To configure the RLE module, use the steps below. A software example is provided at the end of this application note.

- Configure the read eDMA channel, Channel A in [Figure 3](#)
 - a. The channel reading the RLE data must point to the starting address where the RLE data is located and as destination the memory mapped write FIFO.
 - b. 8, 16, or 32 bit transfer size accesses can be used. Configure Source and Destination size accordingly.
 - c. Destination offset will be zero because the write FIFO is memory mapped while Source offset depends on the transfer size.
 - d. The minor loop count depends on the configured threshold of the write FIFO. The minor loop count must be calculated such that it never overflows the write FIFO.
 - e. Set the major loop count to 1 and configure the DMAMUX to trigger transfers each time the write FIFO is below the threshold.
- Configure the write eDMA channel, Channel B in [Figure 3](#)
 - a. The channel reading the uncompressed data must point to the read FIFO where the decoded data is placed and as destination the memory where to output the raw data.
 - b. 8, 16, or 32 bit transfer size accesses can be used. Configure Source and Destination size accordingly.
 - c. Source offset will be zero because the read FIFO is memory mapped while Destination offset depends on the transfer size.
 - d. The minor loop count depends on the configured threshold of the read FIFO. The minor loop count must be calculated such that it never overruns the read FIFO.
 - e. Set the major loop count to 1 and configure the DMAMUX to trigger transfers each time the read FIFO is above the threshold.
- Configure the RLE module
 - a. First, make sure the module is disabled, MDIS=1.
 - b. Set the compressed size of the RLE data in the register RLE_DEC_CISR.
 - c. Set the X and Y original image size in pixels in the register RLE_DEC_DICR.
 - d. Configure the area of interest: set the start pixel coordinates, RLE_DEC_SPCR, and final pixel coordinates, RLE_DEC_EPCR. The area of interest can be the same original image or a smaller part from the original image. Check [Figure 4](#) coordinates example.
 - e. Configure the pixel format, WIDTH, on the register, RLE_DEC_ICR.
 - f. Enable the module so the decoding process starts, MDIS=0.

2.2.1 Decoding non-graphical RLE data

It is possible to decode other types of RLE data, such as sound samples or program code, that is not defined as a 2D image. This type of data doesn't have X and Y attributes, therefore note the following recommendations:

- Select a pixel format for your non-graphical data according the basic format of the data. For instance, if your data is 16-bit sound samples, use 16-bpp.
- If the data decoded size is <65536 symbols, configure X to be the number of symbols and Y = 1 in the register RLE_DEC_DICR.
- If the total data decoded size is >65535 symbols, the encoder must be configured so it generates RLE packets with a maximum size of 65535. Each packet is decoded as an individual RLE data block.
- Set the RLE_DEC_SPCR register to the start point = (1,1).
- Set the RLE_DEC_EPCR register to the final point = (DICR.X, DICR.Y) to decode the complete data chunk.

3 Use case examples

It is possible to use statistical methods to analyze the entropy of certain images and decide whether the compression ratio will be good or not, but typically it is done empirically by just looking at the images. Alternatively, performing an RLE compression of an image is fairly fast in order to apply the trial and error method.

Because RLE is applied along the X axis, as it is how images are natively stored on memory, data having long runs of repeated data on the X axis will be compressed efficiently while data having long runs only on the Y axis will not be compressed.

Some images with good chances to have a good compression ratio are:

- Horizontal gradients
- Any type of shape having a solid background
- Images having single color areas


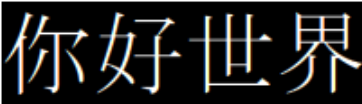

Images coming from the real world as pictures are usually not good choices for compression. The table below showcases some examples of images with its compression ratios.

Table 1. RLE compression examples

Image	Description	Size	Comp. Size	Comp. Ratio
	This image is a round shape with a solid background and single color areas around the center. This type of graphics always results in very good compression ratios. 200x176@8bpp	35200 Bytes	11864 Bytes	3:1
	This is a background image having identifiable black color areas between the green diamonds and further black inside them. This allows a 47% size reduction. 480x272@8bpp	130560 Bytes	69015 Bytes	<2:1

Table continues on the next page...

Table 1. RLE compression examples (continued)

Image	Description	Size	Comp. Size	Comp. Ratio
	This example has a horizontal color gradient with a big black color area; all these characteristics give this picture an excellent RLE compression ratio. 500x400@24bpp	600000 Bytes	111031 Bytes	>5:1
	Raster text is often a good candidate for compression, especially if it is big. 200x57@4bpp (Grayscale)	11400 Bytes	2781 Bytes	4:1
	This is an example of what type of images not to use for RLE compression. The compression ratio is almost unitary. 512x512@24bpp	786432 Bytes	774304 Bytes	1:1

4 Example code

The following example code shows how to configure the RLE decoder module and the eDMA to decode and transfer the data to certain memory location:

```

/* Initialize eDMA channel TCD (0,1) to zeroes */
ptr = &EDMA.TCD[0].SADDR;
for(i = 0; i<16; i++){ *ptr++ = 0; }

/* configure eDMA channel 0 (for write FIFO) */
EDMA.TCD[0].SADDR= (uint32_t)rleBitmap;//Src Bitmap
EDMA.TCD[0].DADDR= 0x90000000 + 0x80;//mem mapped write FIFO
EDMA.TCD[0].SSIZE = 2;// src size 32bits
EDMA.TCD[0].DSIZE = 2;// dst size 32bits
EDMA.TCD[0].SOFF = 4;// src offset 4 bytes
EDMA.TCD[0].DOFF = 0;// dst offset 0 bytes
EDMA.TCD[0].BITER= 1;// major loop = 1
EDMA.TCD[0].CITER= 1;// major loop = 1
EDMA.TCD[0].NBYTESu.R = 28;// minor loop = 28
EDMA.TCD[0].BWC = 3;// BW control helps relief
EDMA.SERQ.R = 0;//enable requests
DMAMUX.CHCONFIG[0].B.ENBL= 1;//enable DMAMUX for channel 0
DMAMUX.CHCONFIG[0].B.SOURCE= 54;//write FIFO request

/* configure eDMA channel 1 (for read FIFO) */
EDMA.TCD[1].SADDR= 0x90000000 + 0xC0;//mem mapped Read FIFO
EDMA.TCD[1].DADDR = 0x60000000;//GRAM mem start

```

```

EDMA.TCD[1].SSIZE = 2; // src size 32bits
EDMA.TCD[1].DSIZE = 2; // dst size 32bits
EDMA.TCD[1].SOFF = 0; // src offset 0 bytes
EDMA.TCD[1].DOFF = 4; // dst offset 4 bytes
EDMA.TCD[1].BITER = 1; // major loop = 1
EDMA.TCD[1].CITER = 1; // major loop = 1
EDMA.TCD[1].NBYTESu.R = 28; // minor loop = 28
EDMA.TCD[1].BWC = 3; // BW control helps relief
EDMA.SERQ.R = 1; //enable requests
DMAMUX.CHCONFIG[1].B.ENBL = 1; //enable DMAMUX for channel 1
DMAMUX.CHCONFIG[1].B.SOURCE = 53; //write FIFO request

/* Configure RLE module */
RLE.MCR.B.MDIS = 1; // disable module
RLE.CISR.B.SIZE = sizeof(bitmap); // compressed size
RLE.DICR.B.X = 240; // pixels/symbols width
RLE.DICR.B.Y = 180; // pixels/symbols height
RLE.ICR.B.WIDTH = 0; // 8bpp size
// decompress full size (240x180)
RLE.SPCR.B.X = 1; // start x pixel (origin)
RLE.SPCR.B.Y = 1; // start y pixel (origin)
RLE.EPCR.B.X = 240; // final x pixel (width)
RLE.EPCR.B.Y = 180; // final y pixel (height)
RLE.MCR.R |= 4; // enable TX FIFO Flush
RLE.MCR.B.MDIS = 0; // enable module

```

The following example code shows how to configure Layer 0 to display a 24 bpp RLE graphic.

NOTE

The following code requires the system and DCU to be up and running. The bitmap data used must be compressed using the RLE MPC5645S format.

```

/* Set DDR Mode ON on the DCU */
DCU.DCU_MODE.B.DDR_MODE = 1; //DDR Mode On
/* Configure the Layer 0 control descriptors */
DCU.LAYER[0].CTRLDESCL1.B.HEIGHT = 272; //Height of layer
DCU.LAYER[0].CTRLDESCL1.B.WIDTH = 480; //Width of layer
DCU.LAYER[0].CTRLDESCL2.B.POSY = 0; //Place this distance from top
DCU.LAYER[0].CTRLDESCL2.B.POSX = 0; //Place this distance from left
// Address of image
DCU.LAYER[0].CTRLDESCL3.R = (uint32_t)rleBitmap; //Compressed bitmap address
DCU.LAYER[0].CTRLDESCL4.B.LUOFFS = 0; //In this case it's the first LUT
DCU.LAYER[0].CTRLDESCL4.B.BPP = 5; //Bits per pixel (= 24)
DCU.LAYER[0].CTRLDESCL4.B.TRANS = 0xFF; //Not transparent
DCU.LAYER[0].CTRLDESCL4.B.BB = 1; //Remove chroma
// Chroma setting
DCU.LAYER[0].CTRLDESCL5.B.CKMAX_R = 0x00; //Max red chroma
DCU.LAYER[0].CTRLDESCL6.B.CKMIN_R = 0x00; //Min red chroma
DCU.LAYER[0].CTRLDESCL5.B.CKMAX_G = 0x00; //Max green chroma
DCU.LAYER[0].CTRLDESCL6.B.CKMIN_G = 0x00; //Min green chroma
DCU.LAYER[0].CTRLDESCL5.B.CKMAX_B = 0x00; //Max blue chroma
DCU.LAYER[0].CTRLDESCL6.B.CKMIN_B = 0x00; //Min blue chroma
/* RLE Configuration registers */
DCU.LAYER[0].CTRLDESCL4.B.RLE_EN = 1; //Enable RLE on this layer
DCU.COMP_IMSIZE.B.SIZE = sizeof(rleBitmap); //Set the compressed image size
DCU.LAYER[0].CTRLDESCL4.B.EN = 1; //Enable layer

```

How to Reach Us:

Home Page:

www.freescale.com

Web Support:

<http://www.freescale.com/support>

USA/Europe or Locations Not Listed:

Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see <http://www.freescale.com> or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to <http://www.freescale.com/epp>.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2011 Freescale Semiconductor, Inc.