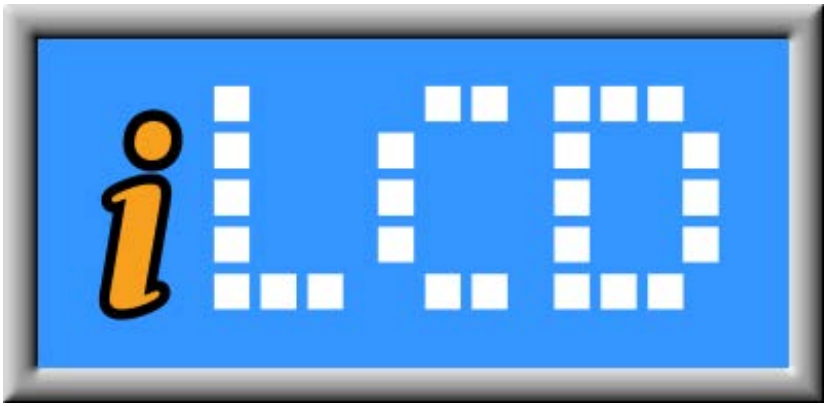

Next Generation Intelligent LCDs

2D Run-Length Encoding Application Note

Version 1.0
Document Date: April 30, 2013

demmel products ■ Radnitzkygasse 43 ■ A-1100 Vienna/Austria ■ Phone +43-1-6894700-0 ■ Fax+43-1-6894700-40 ■ Email: office@demmel.com ■ Web: www.demmel.com

Problem definition

Transferring images via e.g. a serial port from / to a display requires a lot of traffic to be sent over the communication port. To increase speed and decrease the number of bytes to be sent a compression algorithm is mandatory. Standard compression algorithms require a lot of computing effort (in terms of processing power and memory space required) which cannot be afforded by low power embedded systems.

Besides that a scan line based algorithm is preferred to be used enabling the controller to not need to analyze the complete image before compressing it. Speed of compression / decompression is also a very important issue.

Run-length encoding

Run-length encoding is a very simple and fast algorithm to compress single scan lines, especially if the screen does not contain pictures but graphic images used to e.g. control a machine. Run-length encoding is made up of control sequences telling how many repetitions of a single color value will follow.

The disadvantage of simple run-length encoding is that every scan line is compressed separately thus resulting in rather weak performance (in terms of compression rate) especially when processing graphic images. As these images often contain partial identical scan lines following each other a 2D run-length encoding algorithm was developed.

Details of implementation

Control sequences

Control sequences are normally made up by a single byte where the 2 most significant bits indicate the type of repetition and the 6 least significant bits specify the repeat count (a value of 0 specifies a repetition of 1).

Bit 7 (HORZ_REPETITION_FLAG) set only: indicates a (horizontal) repetition of a single color value

Bit 6 (VERT_REPETITION_FLAG) set only: indicates a (vertical) repetition of the same part of the previous scan line

Bit 7 + Bit 6 both set: indicates the start of a 2-byte control sequence where the first byte contains a "repetition increase" ((count + 1) times 64) , specified in bit 0 … bit 5), followed by a second byte containing the type-of-repetition information (horizontal or vertical repetition) and the number of remaining repetitions.

Bit 7 + Bit 6 both cleared: indicates a count of uncompressed data run

Color format used by examples following below

We support 16 bit images only, format is 5 red, 6 green and 5 blue bits.

So the hex value for red is 0xF800, for green it's 0x07E0 and blue has the value 0x001F. Full white is 0xFFFF and full black is 0x0000.

The least significant byte of a color word is sent first.

Example 1

The above image is a 20 x 2 pixel sized image where the first scan line contains one red, green and blue pixel followed by 17 white pixels.

So the first scan line has the following representation:

0x02	3 uncompressed pixels following:
0x00 0xF8	red pixel
0xE0 0x07	green pixel
0x1F 0x00	blue pixel
0x90	indicates a (horizontal) repetition (bit 7 set) of 17 pixel …
0xFF 0xFF	… with white color

The second scan line has 8 identical pixels (compared with the previous scan line) followed by 12 black pixels, so the hex representation is as follows:

0x47	number of identical (bit 6 set) pixels is 8 (vertical repetition) Note: no argument needed in this case
0x8B	indicates a horizontal repetition (bit 7 set) of 12 pixel …
0x00 0x00	.. with black color

Example 2

Example 2 shows how the "repetition increase" sequence is used.

Scan line #1 contains a 68 pixel wide gray scale area followed by a 100 pixel wide blue area. Scan line #2 has a 150 pixel wide area with the same contents as scan line #1 followed by a 18 pixel wide red area. The complete image has a size of 168 x 2 pixels.

Scan line #1:

0xC0	"repetition increase" (bit 7 + 6 set) of 1 x 64
0x03	68 uncompressed pixels (bit 6 + 7 not set) follow: 64 (see above) + 3 + 1
0xFF 0xFF	pixel #0: white color
:	color word-values for pixel #1 … pixel # 66
0x00 0x00	pixel #67: black color
0xC0	"repetition increase" (bit 7 + 6 set) of 1 x 64
0xA3	indicates a horizontal repetition (bit 7 set) of 64 (see above) + 35 + 1 = 100 pixels …
0x1F 0x00	… with blue color

Scan line #2:

0xC1	"repetition increase" (bit 7 + 6 set) of 2 x 64
0x55	bit 6 indicates vertical repetition, count is 2 x 64 (see above) + 21 + 1 = 150 pixels
0x91	indicates a horizontal repetition (bit 7 set) of 18 pixels …
0x00 0xF8	… with red color

Usage of Scan Line Commands

Read 1D/2D Run-Length Encoded Scan Line

```
\i D N r prev_line_offset no_of_pixels
```

Parameter	Type	Range	Description
prev_line_offset	signed byte	-8 ... 8	y-offset of scanline referred to (0 = none = 1D encoded)
no_of_pixels	word	1 ... display width	number of pixel to be scanned

Reads the color information of *no_of_pixels* pixels in a horizontal line from left to right, starting from the current cursor position and reports the data with run-length encoding.

Note

- If `no_of_pixels` + the current column value exceeds the right margin, a *[NACK]* is sent instead of reporting the data.
- When `prev_line_offset` is 0 or `prev_line_offset` + the current row is negative, the data will be 1D encoded.
- When `prev_line_offset` is other than 0 and `prev_line_offset` + the current row is 0 or higher, 2D ecoding with reference to the relatively specified row is carried out.
- Color values in the reported data are made up as 16-Bit Color Values.

Response: `[ACK] no_of_bytes b0 ... [ACK]`

Parameter	Type	Range	Description
no_of_rle_bytes	word	2 ... display width + 2	length of encoded data in bytes
b0 ...	bytes	0x00 ... 0xFF	run-length encoded data

Response Example

```
[ACK][00][03][B1][00][F8][ACK]
```

In this example a line of 50 red pixels has been scanned. The encoded data consists of 3 bytes: 0xB1 indicates a horizontal repetition (bit 7 set) of 50 pixels and 0x00F8 is the 16-bit color value for pure red.

Write 1D/2D Run-Length Encoded Scan Line

```
\i D N w prev_line_offset no_of_pixels no_of_rle_bytes b0 ...
```

Parameter	Type	Range	Description
prev_line_offset	signed byte	-8 ... 8	y-offset of scanline referred to (0 = none = 1D encoded)
no_of_pixels	word	1 ... display width	number of pixels to write
no_of_rle_bytes	word	2 ... display width + 2	length of encoded data in bytes

b0 ...	bytes	0x00 ... 0xFF	run-length encoded data

Writes a horizontal scan line consisting of `no_of_pixels` pixels onto the screen. The `no_of_rle_bytes` long data is sent with run-length encoding.

Note

- The location where the pixels are written to (where the horizontal line begins) is determined by the current cursor position.
- The column address is incremented after every pixel. If the column address exceeds the right margin during printing, *[NACK]* is returned and the command processing returns and waits for the next command introducer (\i). Even so, the scan line is drawn correctly until it hits the screen margin.
- When `prev_line_offset` is 0 or `prev_line_offset` + the current row is negative, only 1D encoded data will be accepted. When a sequence indicating 2D encoding is received in this case, drawing is aborted and and a *[NACK]* is returned.
- When `prev_line_offset` is other than 0 and `prev_line_offset` + the current row is 0 or higher, the data can include sequences for 2D encoding with reference to the relatively specified row.
- Color values in the reported data are made up as 16-Bit Color Values.

Response: [ACK]

Example

```
\iDNw\0\D50\D3\xB1\X1F00
```

This example will write 50 blue pixels to the screen, starting from the current cursor position. The encoded data consists of 3 bytes: 0xB1 indicates a horizontal repetition (bit 7 set) of 50 pixels and 0x1F00 is the 16-bit color value for pure blue.

Revision History

Date	Rev. #	Revision Details
April 30, 2013	1.0	First issue

If you find any errors in this document, please contact demmel products at support@demmel.com