Technical Note

MT9V024
Pixel Defect Correction – Description and Usage

Introduction

Pixel defect correction is intended to compensate for defective pixels by replacing their value with a calculated value based on the surrounding pixels, making the defect less noticeable to the human eye. The locations of up to 32 defective pixels as defined by Aptina can be stored in a ROM on chip during the manufacturing process. Pixel defect correction is not generally recommended for machine-vision applications.

This technical note provides technical details on function and use of pixel defect correction. Sections covered include:

- "Definition of Pixel Defect Correction"
- "Defect Correction Algorithm" on page 2
- "Reading Pixel Defect Addresses" on page 3
- "Limitations" on page 3

Definition of Pixel Defect Correction

The response performance of MT9V024 pixels are qualified under different test condition categories. The number of pixels allowed to vary from the mean for each of these categories is defined in detail in the MT9V024 Outgoing Defect Specification (ODS).

Defect pixel correction is designed to operate on pixels defined as Hot or Very Hot pixels in the ODS:

- A Hot pixel is defined as any single pixel that is greater than 120 LSBs above the mean value of the array when the sensor is operated under no illumination, with an analog gain of 4x and exposure time = 20 msec.
- A Very Hot pixel is defined as any single pixel that is greater than 500 LSBs above the mean value of the array when the sensor is operated under no illumination, with an analog gain of 4x and exposure time = 20 msec.

Note: The total allowable defects of all types, including Hot and Very Hot pixels, is 30.

The locations of these defective pixels are stored in an internal table. When enabled, pixel defect correction uses this table to perform a pixel averaging. A pixel addressed in the table will be substituted with either an average of its neighboring pixels, or its nearest-neighbor pixel, depending on the condition of the neighboring pixels.

The detailed definition of averaging algorithm is provided in the following section.

Defect Correction Algorithm

The correction logic operates on information about five of a pixel's same-color neighbors in the same row. If a pixel is in column C and the readout is not mirrored, its closest left neighbor (L1) is in column (C - 2). L2 is in column (C - 4). Likewise, the nearest right neighbor, R1, is in column (C + 2), R2 is in column (C + 4), and R3 is in column (C + 6). Each of these neighbor pixels can be either good or bad. If a pixel is defined as bad in the chip's ROM, its value will be replaced with a value P using one of the methods below, depending on the condition of its neighbor pixels.

Table 1: Defect Correction Algorithm

L2	L1	P	R1	R2	R3	Notes
	Good	(L1 + R1)/2	Good			
	Good	(3*L1 + R2)/4	Bad	Good		
	Good	(3*L1 + R3)/4	Bad	Bad	Good	
	Good	L1	Bad	Bad	Bad	
Good	Bad	(L2 + 3*R1)/4	Good			
Good	Bad	(L2 + R2)/2	Bad	Good		
Good	Bad	(3*L2 + R3)/4	Bad	Bad	Good	
Good	Bad	L2	Bad	Bad	Bad	
Bad	Bad	(L3 + 3*R1)/4	Good			
Bad	Bad	(L2 + R2)/2	Bad	Good		Uses the corrected value of L2.
Bad	Bad	(3*L2 + R3)/4	Bad	Bad	Good	Uses the corrected value of L2.
Bad	Bad	L2	Bad	Bad	Bad	Uses the corrected value of L2.
n/a	n/a	R1	Good			On the left edge.
n/a	n/a	R2	Bad	Good		On the left edge.
n/a	n/a	R3	Bad	Bad	Good	On the left edge.

Notes:
1. In column mirror mode, the L and R pixel columns are swapped, such that the pixels are still read in the order L2, L1, P, R1, R2, and R3.
2. In any case not covered in the table above, P = 0. (There should be no such cases.)
3. Defective pixels on the left (right in column mirror mode) side of the active window are treated as a special case above. Defective pixels on the right side of the active window can assume that any pixels beyond the edge of the window are defective.
4. In binned modes, each bin is considered to be one pixel for the purposes of the calculations. A bin is defective if any pixel in the bin is defective. Thus, for 2x2 binning, if any of the four pixels in the bin are defective, the bin is defective.
5. Row bin is supported for defect correction; column bin occurs after defect correction, thus, defect correction performs as normal.
6. The datapath must handle up to 12 defective pixels in any given row, in any configuration, after taking binning into account. If a row is programmed with more than 12 defects, or more than 12 defects result from binning multiple rows together, the datapath is not required to correct defects in that row. However, all other rows which have 12 or less defects must still be processed correctly.
7. The datapath is not required to reduce the number of defects after binning. For example, if three rows are binned together, and each had a defect in column 25, the binned row would count as having three defects in column 25. All three defects are in the same bin, and correction is not performed again.
8. Pixel defect correction is supported in the dark rows, which are treated like normal rows for the purposes of defect correction.
9. Pixel defect correction is supported in the dark columns. The dark columns are treated as a separate active region from the real active region. That is, the last dark pixel cannot "see" the first active pixel when calculating its corrected value.

Reading Pixel Defect Addresses

Users may require the pixel locations of the 32-pixel Defect Map.

To read the Defect Map:

1. Disable defect pixel correction: write a value of 0 to R0x07[9].
2. Write the column offset index for the first defect pixel table entry (value is 32 for first pixel, see Table 1) to R0x60.
3. Read R0x60 to obtain the column address of first pixel.
4. Write the row offset index for the first defect pixel table entry (= 33) to R0x60.
5. Read R0x60 to obtain the row address ofthe first defect pixel.
6. Repeat column and row accesses for all 32 entries, or until a read returns value of zero, which means there are no more defect pixel entries.

Table 2: Defect Pixel Location Map

Defect Pixel Number	Column Offset Index	Row Offset Index
1	32	33
2	34	35
3	36	37
4	38	39
5	40	41
...	...	...
29	88	89
30	90	91
31	92	93
32	94	95

Note: Register R0x60 may be read only once after writing an index value. Subsequent reads return indeterminate values.

Limitations

Pixel defect test conditions for the MT9V024 are defined in the MT9V024 Outgoing Defect Specification (ODS). The pixel locations captured during production represent only the defective pixels found under these conditions. Other operating conditions (such as different register settings, or different temperatures) can create a different set of defect pixels. For example, a pixel that may be identified as "hot" under one condition may not be "hot" under another condition and vice versa. Aptina makes no claim to the accuracy of the stored defective pixel data when the device is observed under different conditions. There is no way to change or augment the defective pixel address table at a later time.

Defect pixel correction works well in monochrome sensors. To use defect pixel correction on color (RGB) parts, one needs to enable the color bit (R0x0F[1] = 1) in order for the defect correction algorithm to process nearest same-color neighbors, rather than nearest adjacent neighbors. However, enabling the color bit also results in unequal offsets being added to the three color planes. These unequal offsets in turn are generally not useful to machine vision applications where accurate response is required, and may also distort color high dynamic range mode response.

Revision History

Rev A. 4/29/11

- Initial release

10 Eunos Road 8 13-40, Singapore Post Center, Singapore 408600 prodmktg@aptina.com www.aptina.com

This data sheet contains minimum and maximum limits specified over the power supply and temperature range set forth herein. Although considered final, these specifications are subject to change, as further product development and data characterization sometimes occur.