



Low-Resolution Color Camera Driver for the 8031SDK

Decode a horizontally shuffled
Bayer Colorization Pattern



Introduction

- This project demonstrates how to interface a low-resolution color camera, (equipped with a serial port,) with the 8032SDK

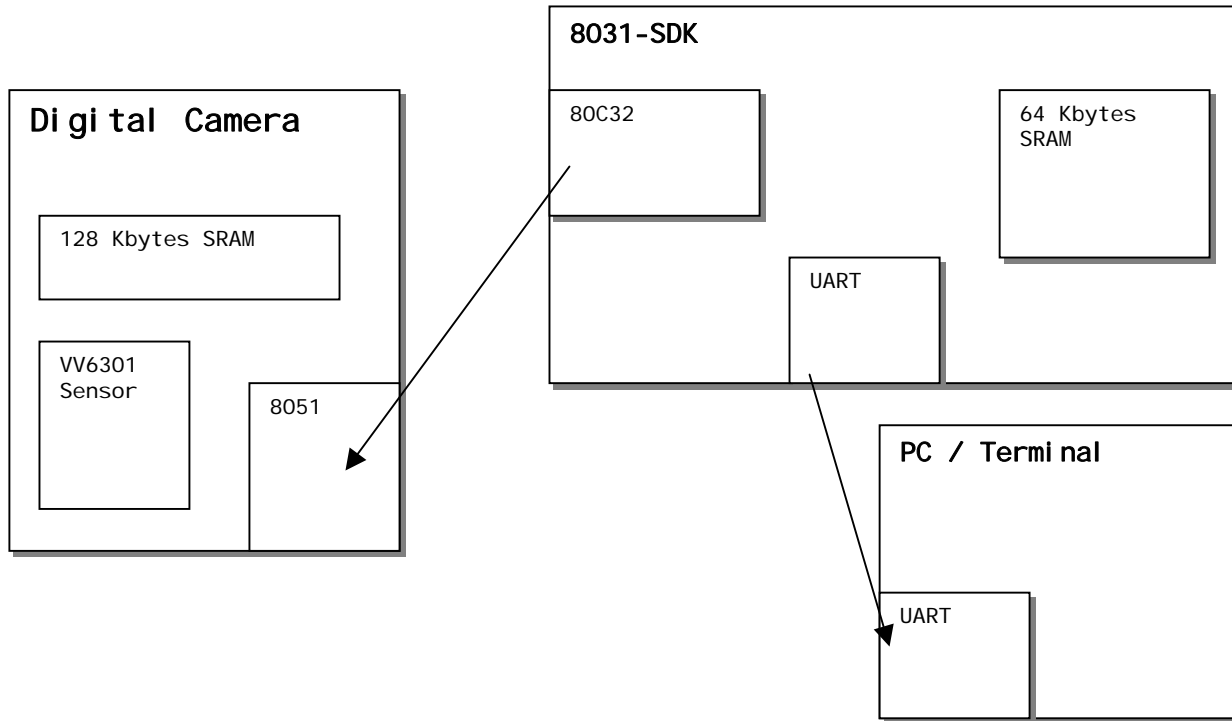


Topics of Discussion



- *High Speed* Serial Communication with the SDK
- Bayer Colorization Pattern
- Bitmap File Structure
- UU-Encoding

Hardware



Serial Communication



- Serial communication between the SDK and the PC through the SDK's external serial port at 9,600 to 38,400 baud. (replace *putchar(char)* and *_getkey()*).
- Timer Calculation for 57,600 baud:
$$11.0592\text{MHz} / (12 * 57,600\text{Hz}) = 16$$

and
$$256 - 16 = 240 = \text{F0h}$$

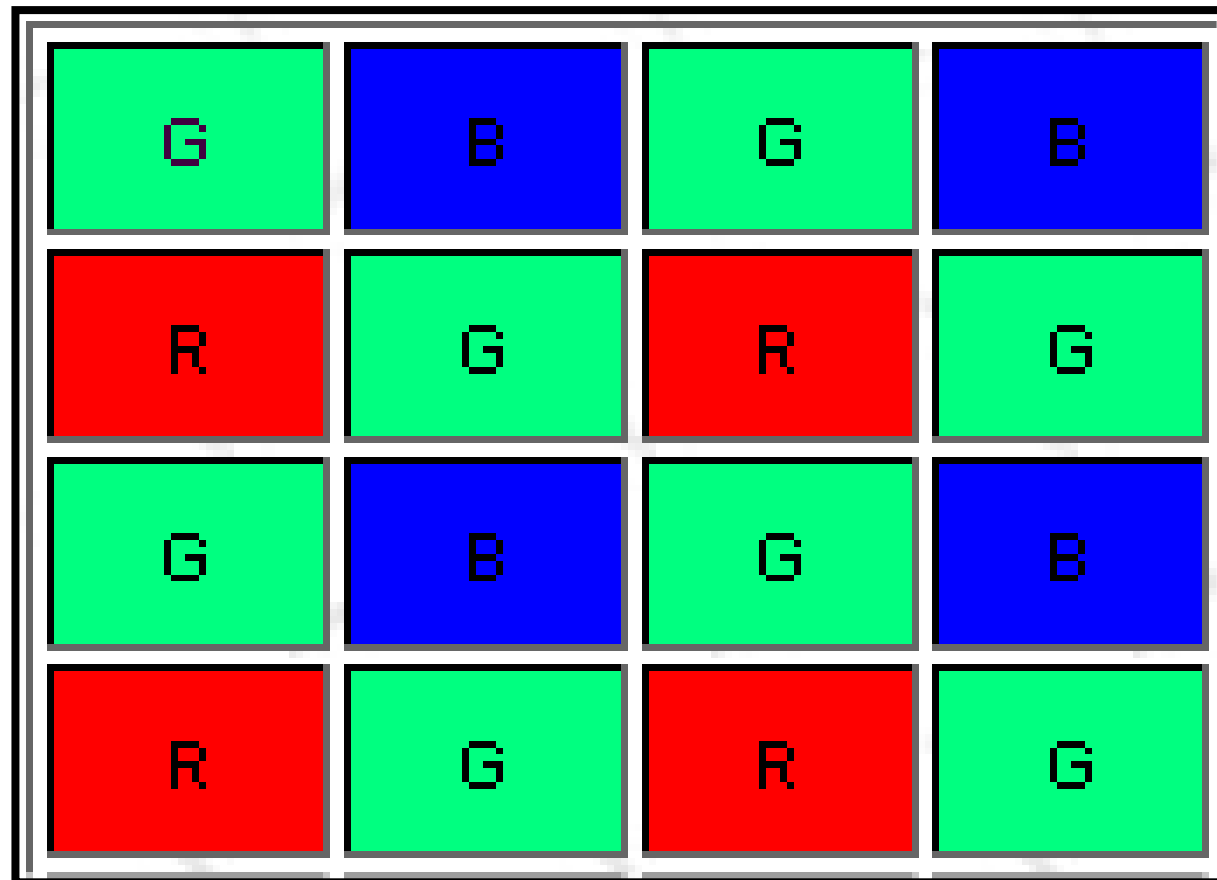
Camera Driver



- A command from the controller has the following format:
 - *STX Command Data 1 Data 2 ... Data N ETX*
- The response has the following format:
 - *STX Response Data 1 Data 2 ... Data N ETX*
- *Example: Grab Image:*
 - *Cmd: 0x02 G 0x03, Response: 0x02 g 0x03*

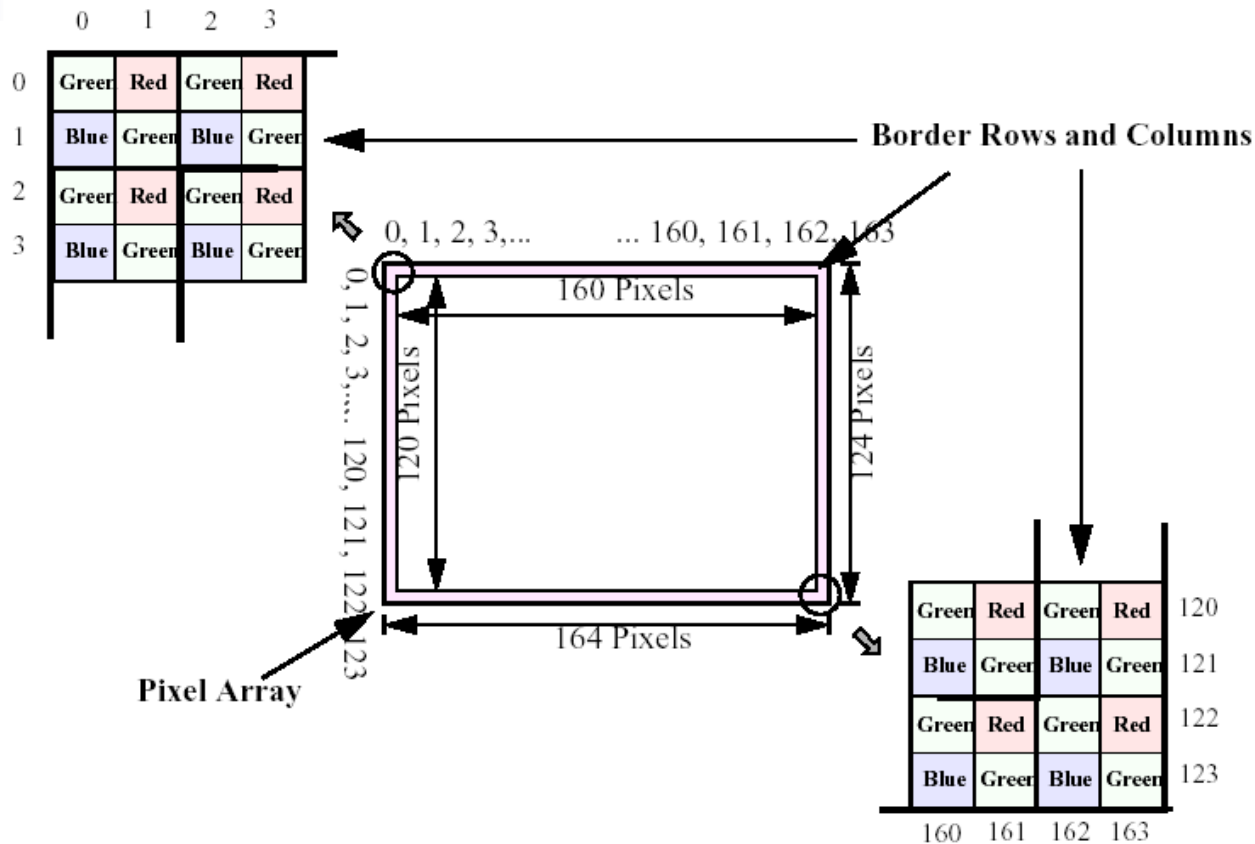


Bayer Pattern



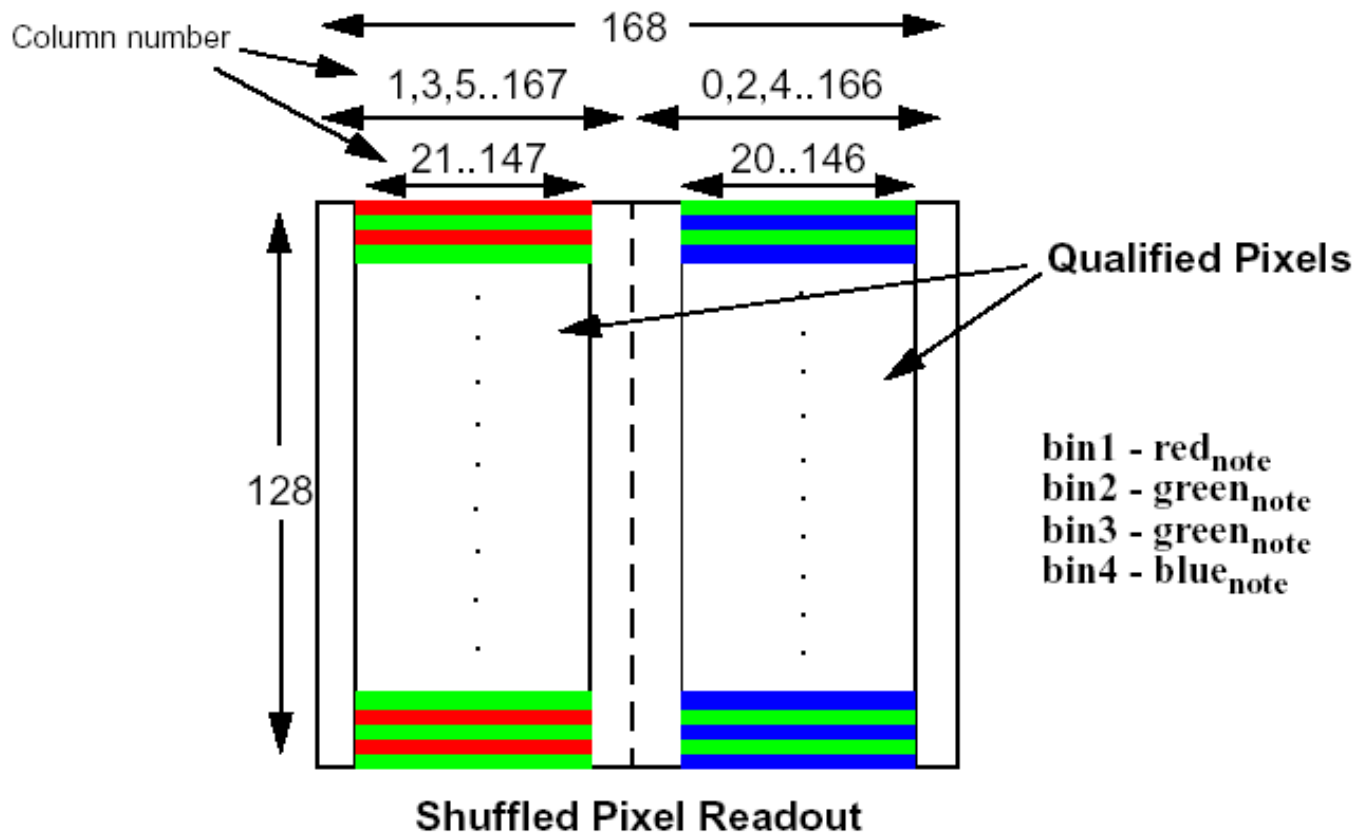


Sensor vs. Data





Horizontally shuffled ...





Interpolation

G1		G3		G5
	G7		G9	
G11		G13		G15
	G17		G19	
G21		G23		G25

Before Interpolation

G7	G7	G9
G11	G13	G13
G17	G17	G19

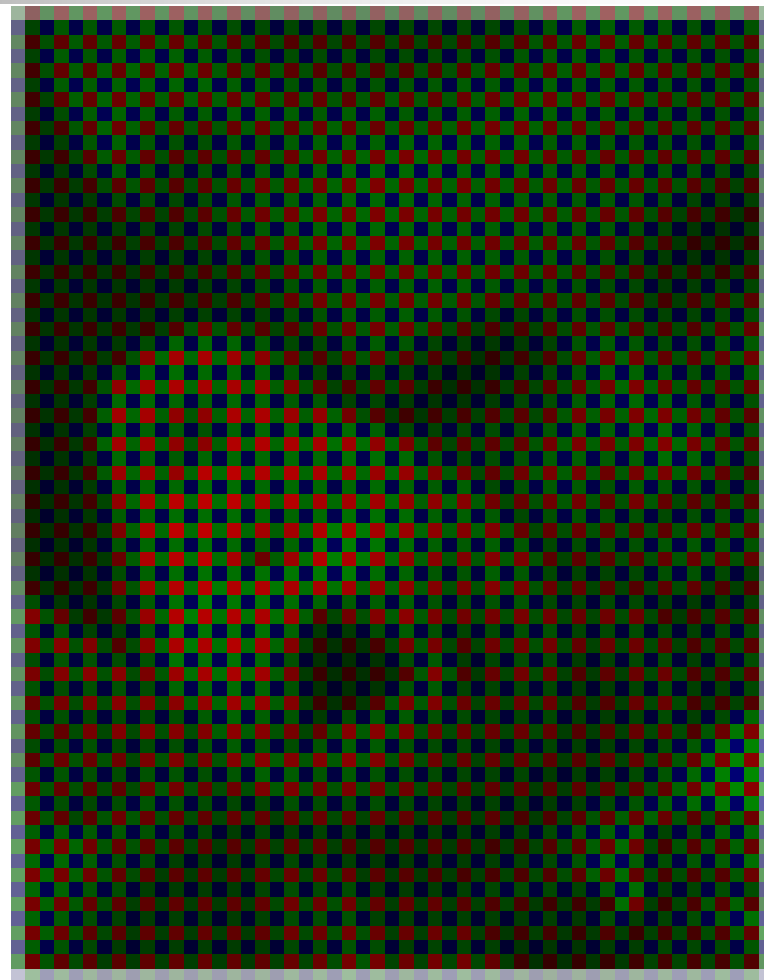
After Interpolation

**Nearest Neighbor
Replication**

**Bilinear
Interpolation:**

G1	R2	G3	R4	G5
B6	G7	B8	G9	B10
G11	R12	G13	R14	G15
B16	G17	B18	G19	B20
G21	R22	G23	R24	G25

Real World Example



Wolf Paulus

ECP II Project

Bitmap Headers



■ Bitmap Header

- ```
typedef struct tagBITMAPINFOHEADER {
 DWORD biSize;
 LONG biWidth;
 LONG biHeight;
 WORD biPlanes;
 WORD biBitCount;
 DWORD biCompression;
 DWORD biSizeImage;
 LONG biXPelsPerMeter;
 LONG biYPelsPerMeter;
 DWORD biClrUsed;
 DWORD biClrImportant;
} BITMAPINFOHEADER;
```

## ■ Bitmap File Header

- ```
typedef struct tagBITMAPFILEHEADER {  
    UINT bfType;  
    DWORD bfSize;  
    UINT bfReserved1;  
    UINT bfReserved2;  
    DWORD bfOffBits;  
} BITMAPFILEHEADER;
```



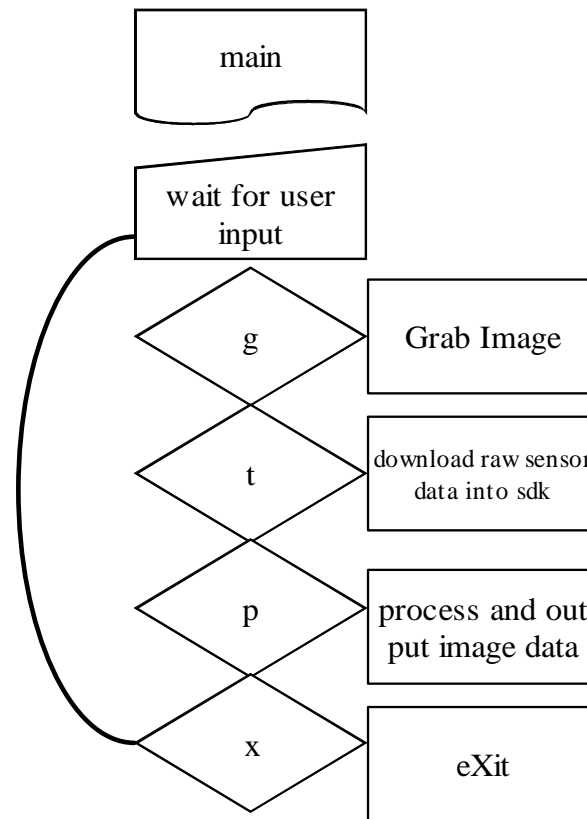
UU Encoding



- Break groups of 3 eight-bit characters (24 bits) into 4 six-bit characters and then add 32 to each six-bit character, which maps it into the readily transmittable character.
- A 6-bit value is in the range $[0..63]$, adding 32 leads to the range: $[32..95] = [0x20..0x5F]$,



User Interface





Demo ...

