Freescale Semiconductor
Application Note

Document Number: MPC5604B
Rev. 0, 01/2011

Using a RCA camera for line detection

Example code based on MPC5604B MCU

by: Francisco Ramirez Fuentes, Marco Trujillo, Cuauhtli Padilla, Rodrigo Mendoza

Contents

1 Introduction

This application note provides details on how a RCA camera works and how it can be used for the specific application of following a line (for the Smart Car Race competition). It includes tips and refers to example code on how to process the signal from the camera on the MPC5604B Freescale microcontroller (MCU); the example code can be downloaded as AN4245SW from http://www.freescale.com.

In this application note, the 1vpp@75ohms output of the RCA camera and the composite video signal output from the camera is explained. The RCA camera used for this application note (CM-26N/P C-MOS – color camera; Figure 2) consist of a pixel matrix array, and a mounted lens of 7.9 mm that provides a field of view equal to subject distance. Some of the advantages of using this camera are:

- No control signals
- Only one output signal
- Complete image view
- Changeable lens
- Auto gain
- Auto contrast
- Auto focused

Besides the above benefits, the only disadvantage is the output signal of the camera is completely analog, which means the user has to be creative to process this signal in order to make it “understandable”. Additionally, this application note provides an example on how to process this signal. Another advantage of this RCA camera is that it is really small and weightless; it is also user friendly since it only needs 3 cables to make it work, supply, ground, and output. These type of cameras are easy to be found through Internet, this particular one was gathered at http://www.sparkfun.com.

NOTE

Rev 0 of this application note provides only the drivers as explained in the example code. A complete example using these drivers will be provided in the next revision of the application note.

2 Camera Signal Interpretation

How is light interpreted? As mentioned before, the camera is a combination of an image sensor and a lens. The light that bounces from the environment enters through the lens, then the lens deflects the light into the sensor. The sensor consists of a microscopic array of capacitors that gain charge depending on light intensity, therefore all pixel charge at the same time and the sensor releases each pixel value in one output signal one after the other until all pixel charges are released. The following image illustrates the process.

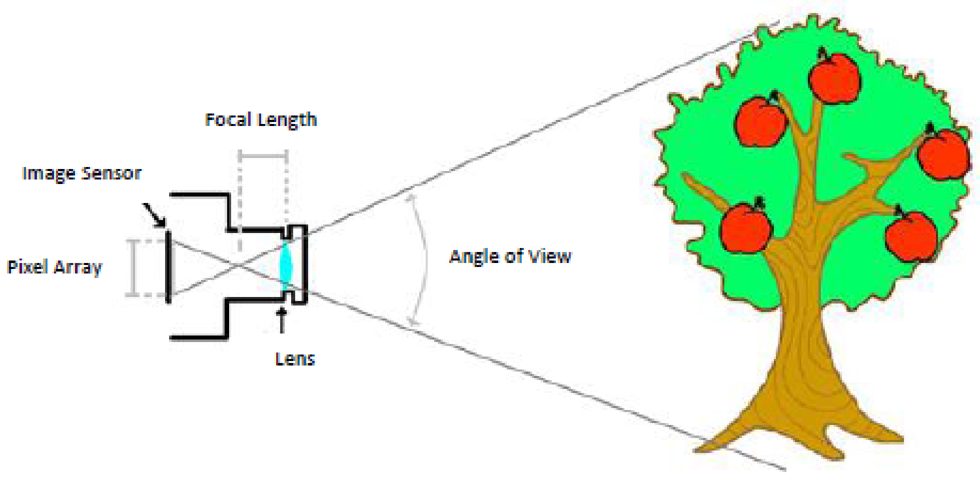


Figure 1. Imaging process in the Lens

The output of a RCA camera is compliant with NTSC standard (not linear), this means the user will be receiving a 60 half images in a second – 30 frames per second (fps). The user needs to consider that the reference code written for this document, is based on a line detection; so, for a complete imaging treatment, extra implementation will be done. The important things to be considered while using this camera are the speed and ease of processing.

The line scan cameras only give the information of a line, RCA cameras show a complete panorama view, with this more complete information users can also consider the behavior of the “line” in the near future, meaning that curves can be early detected. As mentioned before, the user need not worry about lighting conditions as auto gain, compensation and auto contrast are enabled on the RCA camera used for this example. To begin understanding the form of the output of this camera, an easy practice is to connect the output to an oscilloscope (Figure 2).

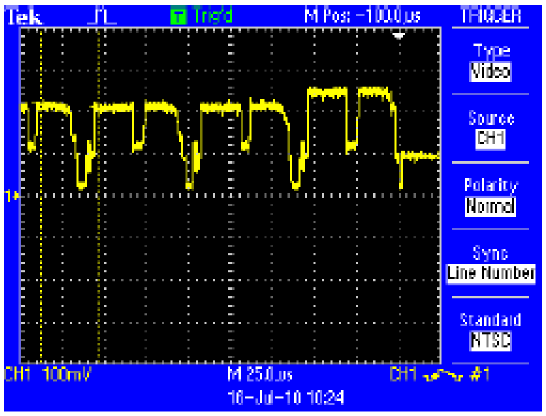
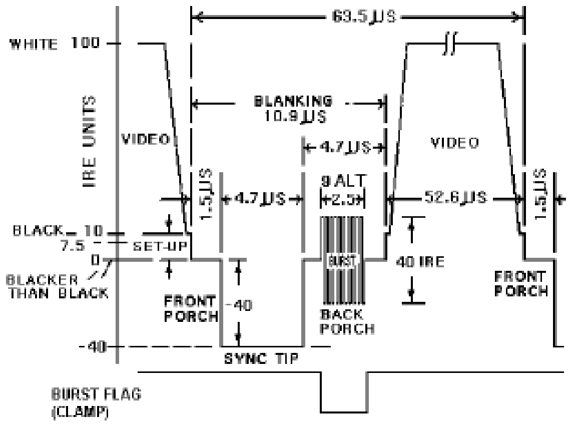


Figure 2. RCA camera and NTSC output signal

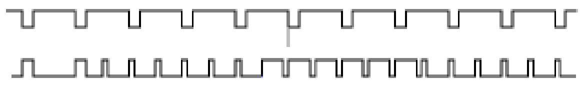
3 Signal Processing

RCA cameras have a general output that almost all standard cameras have; its output is characterized by the -1V to 1V range. The output video signal is heavily composed as shown in Figure 3:

- there is a sync tip used to synchronize the TV with the horizontal video transmission
- after 4.7 µs a color burst is received (this tells the TV which pixels will be filled depending on the color selected by the camera)
- after 4.7 µs, the composed information of the whole pixels
- then the vertical synchronization comes (Figure 3b)



a. NTSC signal



b. NTSC vertical synchronization

Figure 3. NTSC signal description

This application note shows how to process this data with the ADC module. Generally, the faster the better, so the resolution required for your application needs to be considered. For example; if conversions are made in 10 µs, it may not be fast enough to have a good resolution for this application, since the video part of the signal last 52.6 µs, it means this data conversion will be 5 pixel resolution.

Generally, it will be useful to have 20 pixels resolution or more, for some applications 10 pixels will be enough (low resolution applications), this means that the MCU will be able to sample at less than 3 ms to have a good image resolution. Other important things to be considered is the ADC resolution. The 10-bit ADC of the MPC5604B MCU helps to make a 40 pixel resolution image.

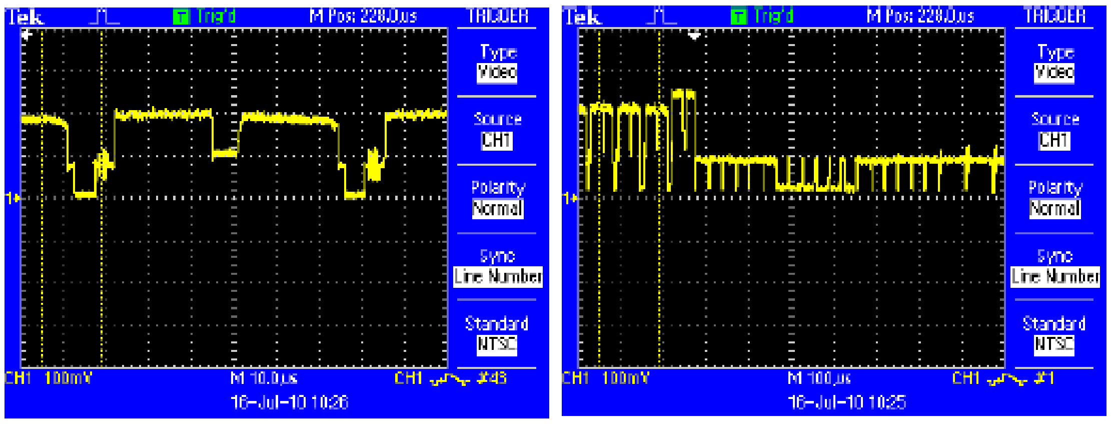
It is important to understand that the following have been considered to read the video signal:

- If the ADC is fast enough to read the 4.7 µs sync signal then it can process all of it with the ADC module
- User should be able to recognize the synchronization pulses to build the pixel matrix, and know when to begin the next capture of the image

Since in this application note, the idea is to recognize a “black line” on a white background, user does not need to worry on decoding the color signal since every color will send the same information as black has no light interpretation and white is composed of all colors. After this, the main process is to wait 4.7 µs after the horizontal sync to trigger the ADC capture.

Every time the MCU sees the vertical sync (Figure 5a) it must be able to recognize it with the ADC, so that the next incoming video signals are the beginning of a new image, and it should start at beginning of the matrix. Every time it sees a horizontal sync it means one video line has finished and it must start capturing the next video line.

The time it has to process the gathered matrix is in the vertical sync period, at this time there is no live video streaming so the MCU can do all the required process in this space.

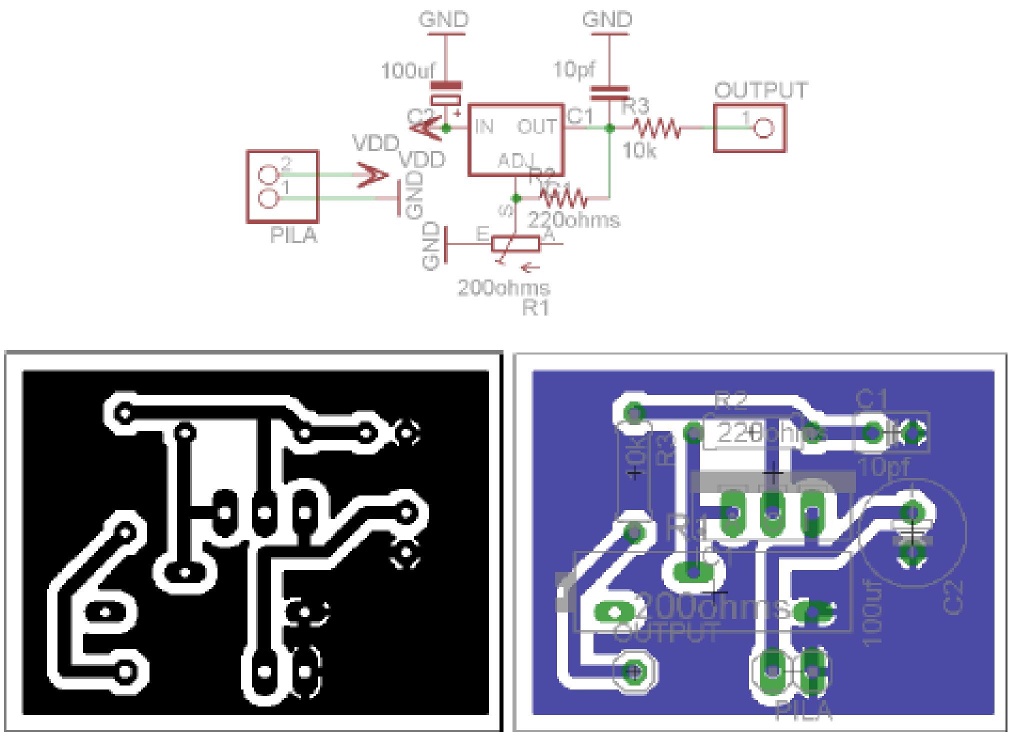


a. Video Signal b. Vertical Sync Signal

Figure 4. Video signal and vertical synchronization signal

4 External components required for the RCA camera

The voltage levels required for the MPC5604B are between 3.3 and 5.0 volts. So having the video signal running between these levels is needed, and it is simply achieved by injecting a voltage to the signal. Figure 4 presents a simple schematic to make the signal rise up between the desired voltage levels; user can select a static regulator or a variable one. A variable voltage regulator is recommended as it will manipulate the output more easily. In this example the variable regulator is injecting 1.3 volts with a strong pull-up (10KΩ) into the output, and this assures the incoming signal goes from 0 V to 2.0 V. So the strong pull-up assures there is less consumed current and protects it from a short circuit. After setting this and assuring it works, user should be able to see NTSC signal (Figure 3a).



a. Layout b. Component Placement

Figure 5. Suggested schematic to handle NTSC signal

5 Software/Driver design and description

Interfacing MPC5604B with a RCA camera

Configurations set to interface MPC5604B to the RCA:

- core running @64 MHz
- ADC module in scan mode – 1MHz speed for one conversion every 1µs
- Interrupts with the end of conversion on the ADC module
- ADC watchdog to increase performance

For capturing pixel values and developing useful data, the code has been developed to capture an entire image (40x250 pixels) at a rate of 60 fps; there is a space between each taken frame so the user can process the gathered image data.

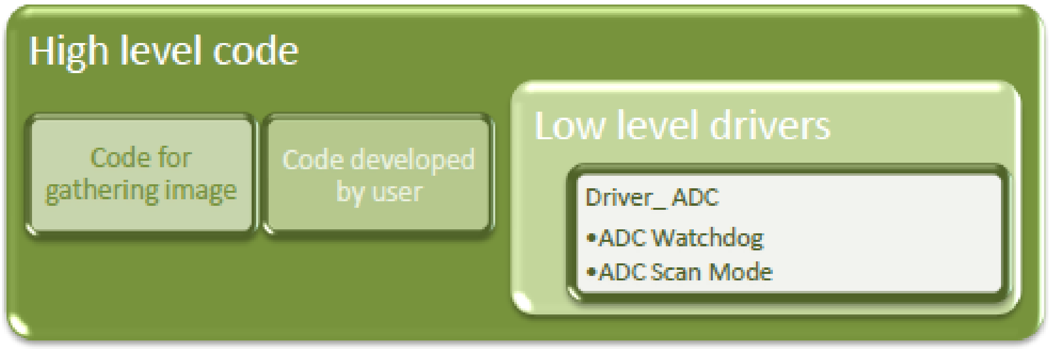


Figure 6. APIs implementation

According to Figure 6, user simply can decide how frames data will be processed. As the frame was already gathered, the *u32picturematrix[X]* variable is a matrix of 1x246, it contains the position of the line from 5 to 51, of every line of pixels, this means that 51-5 = 46 pixels. This means we have 46 pixels for every line, and we have 246 lines available which gives us a 246x46 pixel resolution.

For example (Figure 5a), if *u32picturematrix[123]=28* , then this means that right in the center of the frame is being captured a black color (black line centered; 246/2 = 123, 28 = black color).

This matrix (Table 1) provides us all the information of the picture we need, if we receive a 0 in the function it means, it has acquired an invalid value; which will happen when there was no line to detect or the camera is observing just a white background with no line to follow.

To obtain data you should go and process directly the *u32picturematrix* since the entire image taking has been already done. In the example code, you can find many lines are taken so the user can detect the position of the black line against the position of the camera.

Table 1. Low level functions

Variables involved	u32picturematrix	Global array used to save the picture taken; it gives you information of the position of the line in 246 lines.
	u32pixeloverflow	Global variable used for detecting which is actual pixel
	u32startpulse	Global variable used to know when and where are the sync pulses.
	u8done	Global variable used to know when it can process the taken image.
	u32imageline	Global variable used for example code.
	u32index	Global variable used as an index to know in which is the actual pixel line.
	u32debuggingpurposevariable	Global variable used for debugging purposes.
	u32min	Global variable used to know min position value of the line
	u32posmin	Global variable used to know position of the line.
	u32difpicture	Global variable used to validate taken data.
	u32sum	Global variable for the example.
	u32cont	Global variable used to know if image overflows.

6 Conclusion

Table 2. Advantages and disadvantages of this method

Advantages	Disadvantages
• No external analog processing needed • Never faces problems for calibrating • Can “see” in dark places	• Cannot accelerate frame rate • You have less time to process the image since almost all the time is taken to gather it.

This method is comfortable as external processing/filtering is not necessary and there is no calibration problem. The only problem can be the excessive amount of information gathered that needs to be processed by the ADC. So the architecture of the application needs to consider this amount of throughput.

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

Document Number: MPC5604B
Rev. 0, 01/2011

freescale™
semiconductor