Freescale Semiconductor
Application Note

Document Number: MPC5604B
Rev. 0, 01/2011

You may find useful up to page 4

~Ihab

Using Parallax TSL1401-DB Linescan Camera Module for line detection

Example code based on the MPC5604B MCU

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

Contents

1 Introduction

This application note explains how the Parallax TSL1401-DB camera works and how it can be used for the specific application of following a line for the Smart Car Race competition. This document includes tips and refers to the example code on how to process the signal from the camera on the MPC5604B microcontroller (MCU); the example code can be downloaded as AN4244SW from https://www.freescale.com. The Parallax TSL1401-DB LineScan Camera Module consist of a CMOS linear sensor of 128 pixel array and a mounted lens of 7.9 mm, these provide a field of view equal to the subject distance. Some of the advantages of using this camera are the following:

- Simple and easy to use
- Frequency of capture can be manipulated by the user
- Enough definition for line following application
- Removable and interchangeable lens for different resolutions

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”. This document provides two different methods to process this signal.

There are many evaluation boards for the Parallax camera, which can be another inconvenience. For this case, a benchmark was done and “Parallax BASIC Stamp DB-

freescale semiconductor

Expander Daughter board-to-SIP" or the "Parallax Stamp 2pe Daughter board Extension Cable" was found. These were found as the easiest to use from https://www.parallax.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 (linear in this case) and a lens. The light that bounces from the environment enters through the lens, and the last one deflects 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.

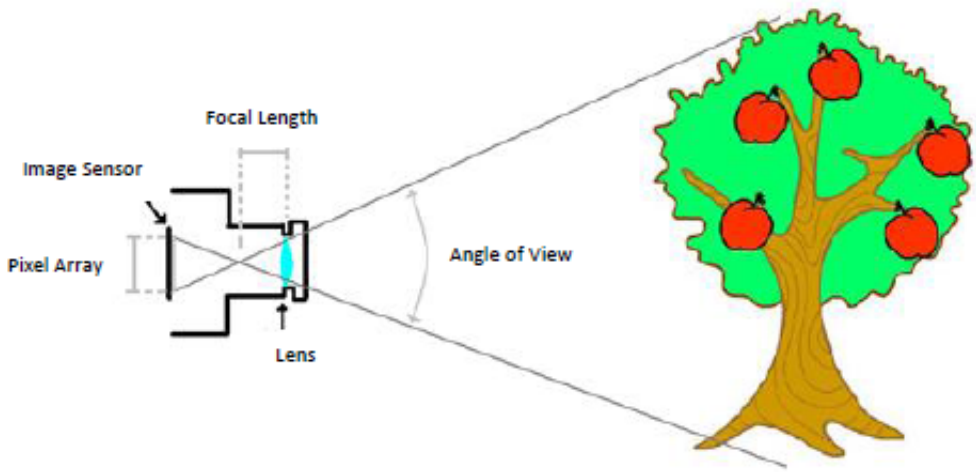

Figure 1. Imaging process in the lens

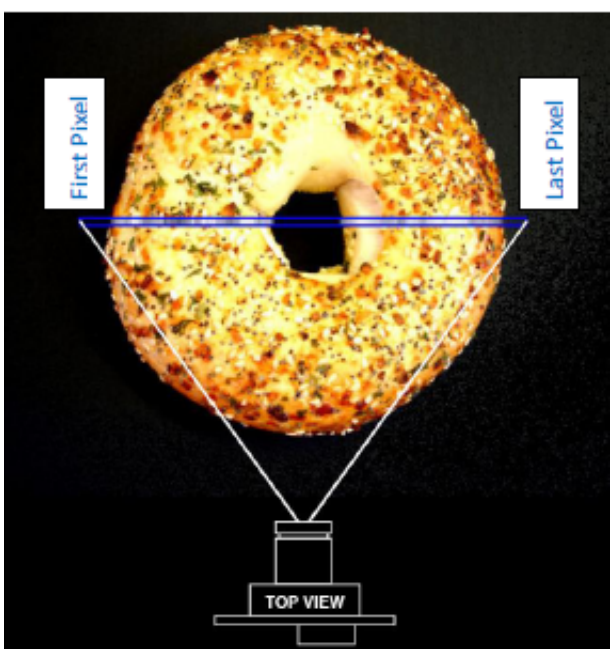

Figure 2. One caption example

Because the camera is using a linear sensor it is impossible to gain a full view of the panorama in a single shot, therefore it only takes one line of the full panorama as shown in the next image. Here, the line to be captured is completely dependent on the distance to the lens as shown in Figure 2. Finally this image is released in an analog signal as shown in Figure 3.

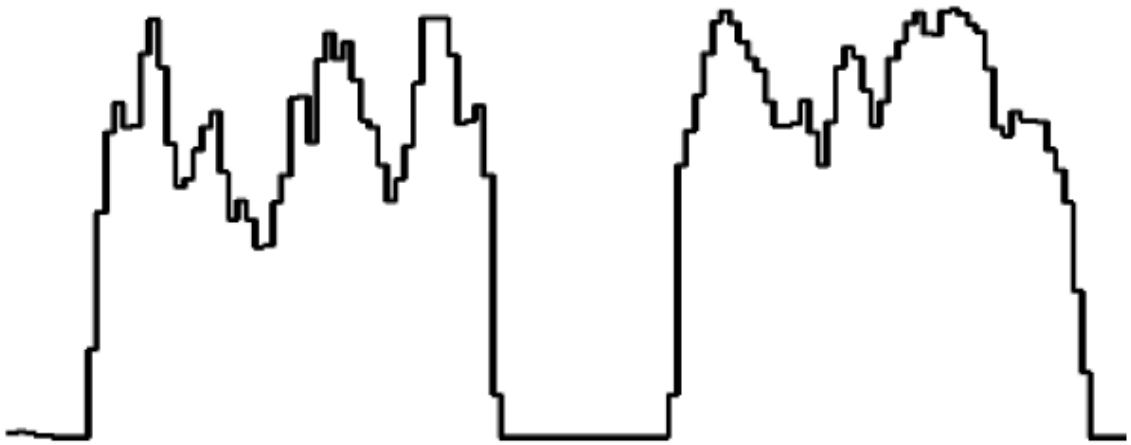

Figure 3. Output signal from parallax camera

3 Signal processing

For normal operation of the camera, the user needs to take care of the following signals only:

- CK (clock)
- SI (serial input)
- AO (analog output)

where CK and SI are camera inputs and AO is a camera output.

As mentioned before, the user can manipulate the frame speed of the camera by adjusting CK and SI signals. Faster is the CK frequency, the faster the camera releases the pixel values, and the closer each SI is from one another the faster each frame capture occur. It is important to understand, the faster the frame capture occur the lower each pixel gain charge. This leads to another important factor, the “integration time”.

Integration time (Figure 4) is the time the pixels have to complete its charge. With very long integration time, the pixels will be saturated even if there is low light intensity in the environment, on the other hand with very short integration time the pixel will not gain charge even if there is excessive light on the environment.

As it can be seen in Figure 4, after 18 clock cycles, the pixels begin to charge back again. After 129 clock cycles all pixels are released from the camera, this means, from that moment it can send another SI pulse to release AO signal again; but as mentioned before longer the cycle, the pixels will charge more and user will get better pixel quality.

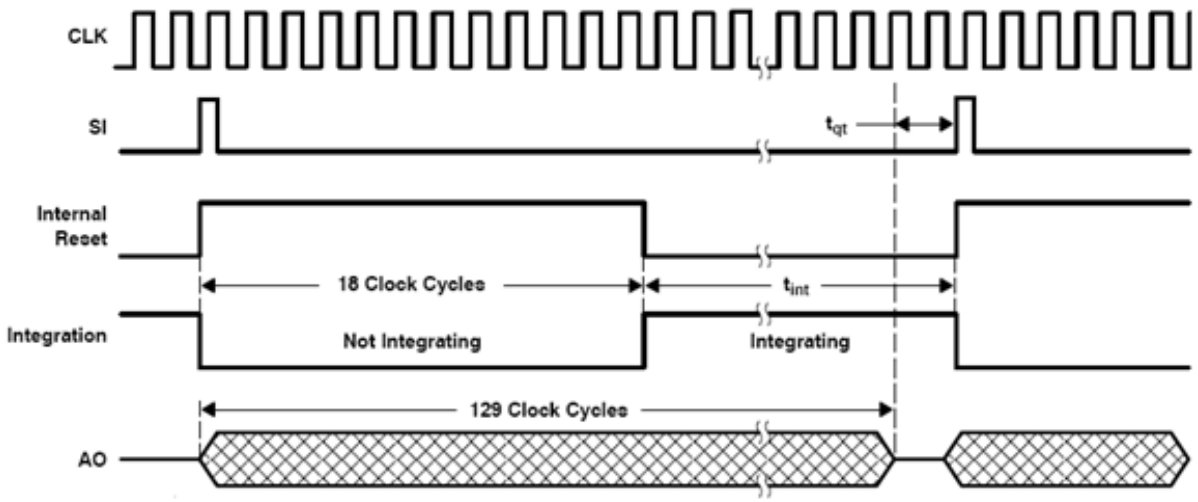

Figure 4. Integration time

These three signals (CK, SI and AO) are synchronized (Figure 5), so the SI and CK signals will be adjusted to start the camera operation correctly:

1. Both signals must have the same pulse width
2. Both must be perfectly aligned out of phase for half pulse
3. As a result AO signal will be perfectly aligned with CK.

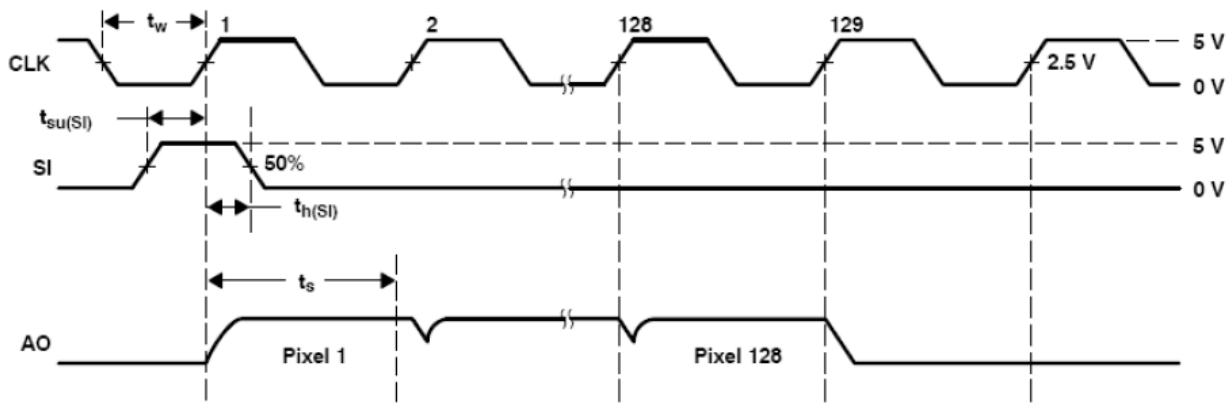

Figure 5. SI and CK signals synchronized to get AO

NOTE

Once all the 129 clock cycles are passed, it is very important to shut down the CK signal, this will help the pixels to charge much better than if CK signal is on forever. For more information about the basic operation of the camera module look at: http://www.parallax.com/Portals/0/Downloads/docs/prod/acc/TSL1401-DB_manual.pdf

4 Software/Driver description

SI and CK input pulse implementation

Generation of SI and CK input pulses is implemented through Driver_Linear_Cam.c.

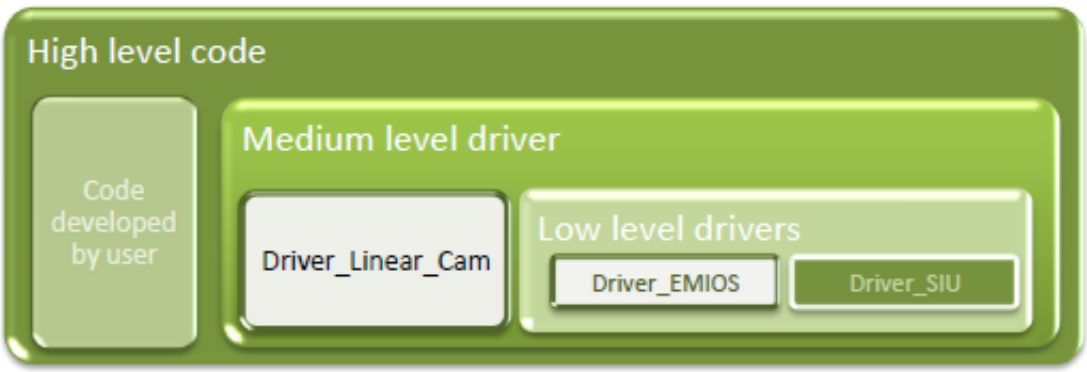

Figure 6. APIs implementation (version 1)

According to the block diagram in Figure 6, the user will have to call functions from the *"Driver_Linear_Cam"* driver, which at the same time calls functions from low level drivers, that corresponds to functions from the *Driver_EMIOS.c* and *Driver_SIU.c* drivers. In these medium and low level APIs, functions described in Figure 7 can be found.

In these drivers implemented, there are default parameter values for the low level functions which will work for launching the Parallax linear camera.

It is useful to know that the user can change the value of the channels to be used at the top of each driver in the *definitions* section and the value of the *Path Configuration Register*, defined in *Driver_MPC5604B.h* for most of the eMIOS and ADC channels.

Integration time can also be manipulated by the second parameter in the function *vfnInit_EMIOS_0_Mcb(CAM_CNT_CHAN, INTEGRATION_TIME)* , where the *INTEGRATION_TIME* means that the SI pulse will wait for *INTEGRATION_TIME* times, for the internally pre-scaled clock cycles to start again.

After finishing your configurations, call for the "vfnInit_All()", "vfnSetup_CamLin()" and "vfnInit_EMIOS_0" functions in your main and the pulses will be generated.

NOTE

It is recommended to check the pulses in an oscilloscope before putting them directly to the camera to make sure signals are as specified.

Table 1. Driver_Linear_Cam: vfnSetup_Camlin() function

Function	vfnSetup_CamLin	Setup configuration for Camera: define modulus counters for clock and start pulses, as well as Opwm channels for each one.
Parameters		None
Return		Null

Table continues on the next page...

Table 1. Driver_Linear_Cam: vfnSetup_Camlin() function (continued)

Functions from SIU	vfnInit_Emios_Output_Pad	Initialize a microcontroller pin as an output for an EMIOS channel.
Functions from EMIOS	vfnSetup_Emios_0	Enable Emios clock, configure preescaler to generate 4 MHz Emios clock, enable global time base, enable Freezing channel to freeze them when in debug mode.
	vfnInit_Emios_0_Mcb	Define Emios channel as Modulus up counter buffered with period selected by the A parameter, configure preescaler to produce 1 MHz time base.
	vfnInit_Emios_0_Opwm	Define Emios channel as positive Opwm with time base corresponding to the counter bus B,C,D,or E and establish its raising and falling edge with parameters A and B.
Definitions involved	CAM_CNT_CHAN	Emios channel used as Modulus Counter for SI signal.
	CHANNEL_CK	Emios channel used to generate CK signal for camera.
	CHANNEL_SI	Emios channel used to generate SI signal for camera.
	PCR_EMIOS_0_tag	Pad configuration register required to configure SIU module. Defined in “Driver_MPC5604B”.

Focusing the camera

Once the sensor is perfectly working the next step is to find the best position of the lens that will generate the clearest images. The best way to do it is using an oscilloscope (Figure 8);

1. Connect the SI and AO signals to the oscilloscope
2. Set the SI pulse so that it can be clearly seen and then trig the AO signal with the SI signal using the trig function
3. Fix the camera looking at a sheet of paper with a black line in the center
4. The image of the black line will appear on the oscilloscope screen
5. Screw the camera until you find the position where the line seems the clearest

It is possible to change the lens of the camera. To search for the correct lens look for them as “Board camera lens” you will find the angle of view is measured in millimeters, which corresponds to the focal length. The angle of view depends on the focal length and the image plane (Figure 9); for the Parallax image sensor the image plane is of 1/3” so be sure to take this last number in count while making your decision.

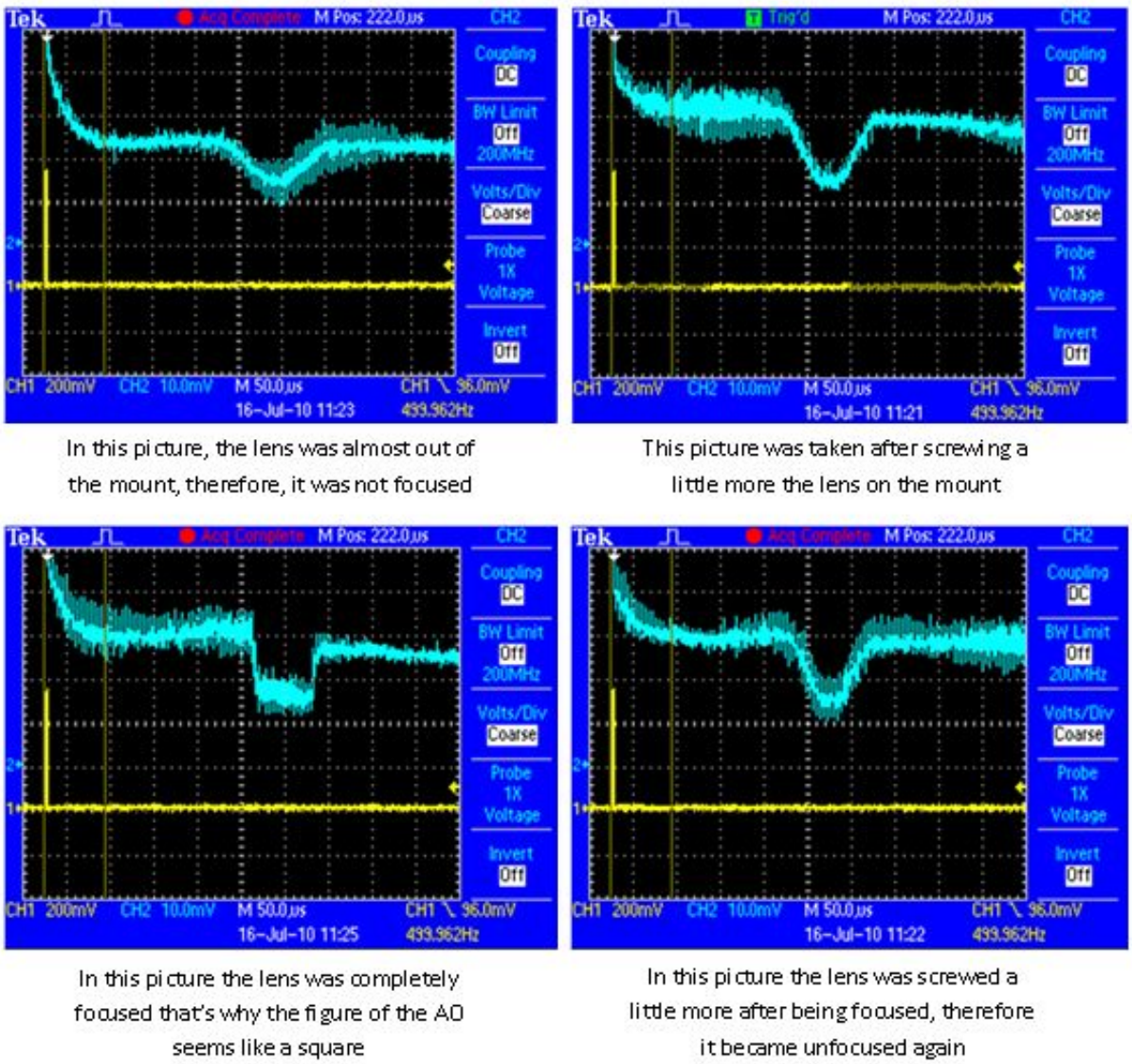

Figure 7. Scanning a black line

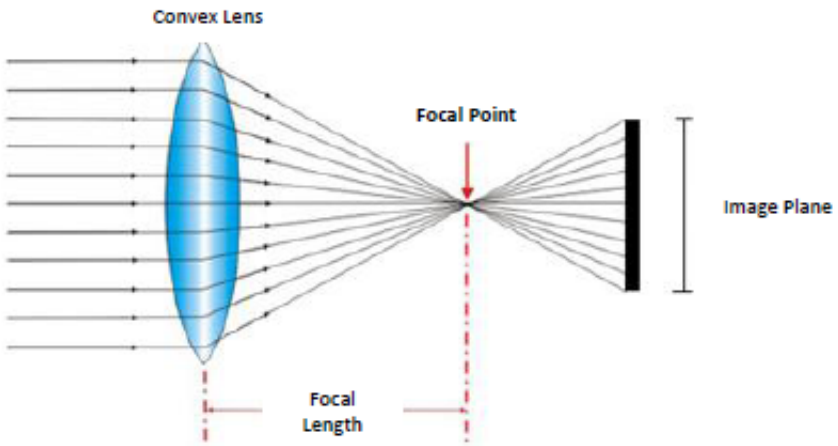

Figure 8. Lens structure

5 AO signal processing implementation

It is implemented using the eMIOS as IPWM (Input Pulse With Measurement). The first method explained in this guide consists of manipulating the AO signal of the camera to convert it into a pulse signal. The idea is to filter the AO signal with some external components in order to produce a readable signal for the eMIOS IPWM mode (Figure 9). The second method explained in this guide consists in generating another eMIOS OPWM (Output Pulse Width Modulation) with the same period of the clock pulse but out of phase for one fourth of its period. The purpose of placing this signal out of phase is to generate the "cross triggering" exactly where the pixel can be read (Figure 10). To understand this behavior better look at the following image.

1st Method: External filtering and eMIOS IPWM

The external filtering procedure consists of introducing the AO signal into a fast comparator, establish a threshold with a potentiometer in the comparator, the resulting signal will oscillate between 0 and 5 volts. Then introduce this last signal into a flip-flop with a clock signal that is equal to the one that enters the camera but this time inverted (Figure 9). The resulting signal consists of two positive pulses and two inverted pulses. The smallest inverted pulse width corresponds to the line width and the positive pulses correspond to the rest of the 128 pixels. The large inverted pulse is the extra integration time. Now, with this filtered signal it can be used in the IPWM mode of eMIOS module to do the rest of the caption; the resulting signal must be connected to an eMIOS channel that can be configured as IPWM. The recommendation is to connect the signal to any channel that has the same modulus counter as the SI pulse (refer to MPC5604BRM; 23.6.1.1.4 Input Pulse Width Measurement (IPWM) Mode). To interpret this input signal with the MC5604B MCU, it can use functions *vfnInit_CamLin_Ipwm()* , *Set_Line_Width()* and *u8Capture_Line_Values()* of the *Driver_Linear_Cam.c* API (Figure 6).

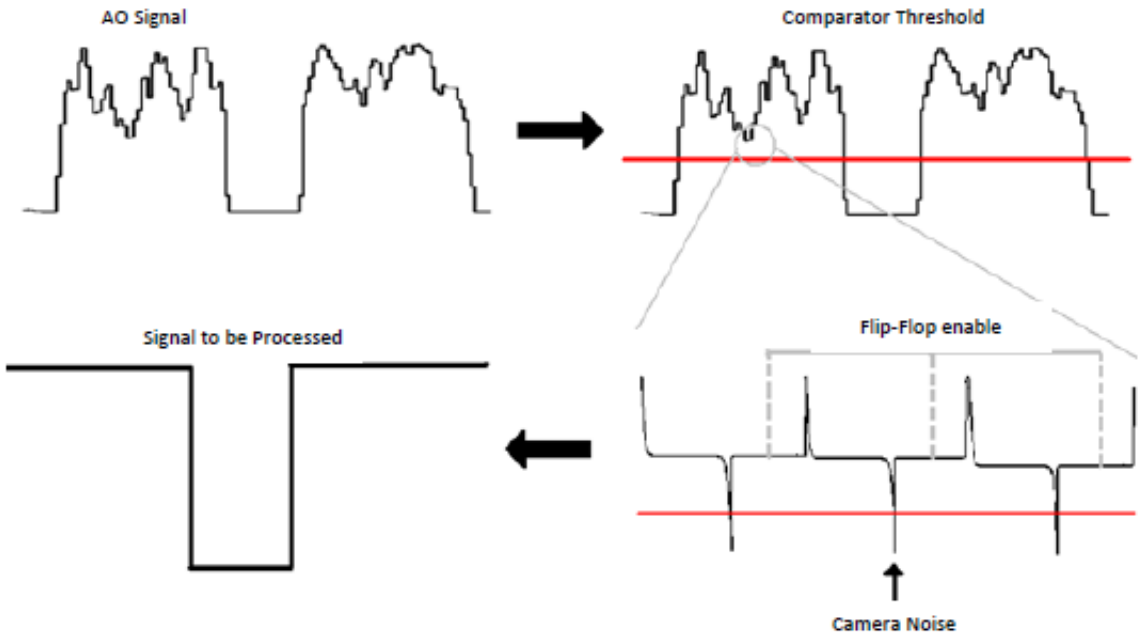

Figure 9. AO external filtering

Table 2. Driver_Linear_Cam: u8Capture_Line_Values() function

Function	u8Capture_Line_Values	Captures the value from the first pixel position to the line position, as well as the line width depending on the filter done by TolWidth and TolDelta (percentage).
Parameters	u8TolDelta	Tolerance (percentage) admitted for the next delta x value in comparison to the previous in order to be considered as a valid value.
	u8TolWidth	Tolerance (percentage) admitted for the next line width value in comparison to the established average value in order to be considered as a valid value.
Return	1	Successful value recognition.
	0	Unsuccessful value recognition.
Functions from EMIOS	EMIOS_0.CH[#].CSR.B.FLAG	Register of the status of the selected channel flag.
	EMIOS_0.CH[#].CADR.B.CADR	“A” register from selected channel of the EMIOS module.
	EMIOS_0.CH[#].CBDR.B.CBDR	“B” register from selected channel of the EMIOS module.
Definitions involved	CAM_IPWM_CHAN	EMIOS channel used as IPWM to capture line pulse.

Table continues on the next page...

Table 2. Driver_Linear_Cam: u8Capture_Line_Values() function (continued)

Variables involved	u16center	Global variable used to save the value of the position of the line center before filter.
	u16width	Local variable used to save the value of the line width before filter.
	u16LineWidthMeasure	Global variable used to compare actual line width with established average line width for filtering.
	u16LineWidth	Global variable used to save the value of the line width after filter.
	u16Center	Global variable used to save the value of the position of the line center after filter.

Table 3. Driver_Linear_Cam: Set_Line_Width() function

Function	Set_Line_Width	Define the value of an average line width from selected samples, define the actual delta X value as a result of an average of the same samples.
Parameters	u8Frames	Number of desired samples to calculate variables.
Return		Null
Functions from EMIOS	EMIOS_0.CH[#].CSR.B.FLAG	Register of the status of the selected channel flag.
	EMIOS_0.CH[#].CADR.B.CADR	“A” register from selected channel of the EMIOS module.
	EMIOS_0.CH[#].CBDR.B.CBDR	“B” register from selected channel of the EMIOS module.
Definitions involved	CAM_IPWM_CHAN	EMIOS channel used as IPWM to capture line pulse.
Variables involved	i	Local variable as counter for capturing requested number of frames.
	u16LineWidthMeasure	Global variable that gets the value of the average width of the line.
	u16Center	Global variable that gets the value of the average position of the center of the line.

Table 4. Driver_Linear_Cam: vfnInit_CamLin_Ipwm() function

Function	vfnInit_CamLin_Ipwm	Set global variables to 0 and define an Emios channel as IPWM with selected polarity.
Parameters		None

Table continues on the next page...

Table 4. Driver_Linear_Cam: vfnInit_CamLin_Ipwm() function (continued)

Return		Null
Functions from SIU	vfnInit_Emios_Input_Pad	Initialize a microcontroller pin as an input for an EMIOS channel.
Functions from EMIOS	vfnInit_Emios_0_Ipwm	Define Emios channel as IPWM with selected polarity and time base corresponding to the counter bus B,C,D,or E.
Definitions involved	PCR_EMIOS_0_tag	Pad configuration register required to configure SIU module. Defined in “Driver_MPC5604B”.
Variables involved	u16LineWidthMeasure	Global variable set to 0.
	u16Center	Global variable set to 0.
	u16LineWidth	Global variable set to 0.

To obtain the values of line width and position of the line for the high level algorithm it is necessary to call functions *u16_LineWidth()* and *u16_Center()*. These are just return functions for the *u16LineWidth* and the *u16Center* variables.

Table 5. Advantage and disadvantage of external filtering and eMIOS IPWM

Advantage	Disadvantage
Fast and easy process for the MCU	Needs external analog processing
ADC module is free for other applications	Have many calibration conflicts

2nd Method: Processing with ADC and CTU

For this method, CTU (Cross Triggering Unit) will be used. As the CTU is in charge of triggering between eMIOS and ADC modules, it can start an ADC conversion when a eMIOS module flag is activated. As mentioned before; it is being generated as eMIOS OPWM with the same period of the clock pulse but out of phase for one fourth of its period. The purpose of placing this signal out of phase is to generate the “cross triggering” exactly where the pixel can be read (Figure 10). The CTU signal will trigger the ADC conversion when the pixel charge is in good conditions to be read, for this reason it is important to align these signals as shown in Figure 10. For capturing pixel values and developing useful data, use the *vfnInit_CamLin_Adc()* and *u8Capture_Pixel_Values()* functions of the *Driver_Linear_Cam.c* driver (Figure 11).

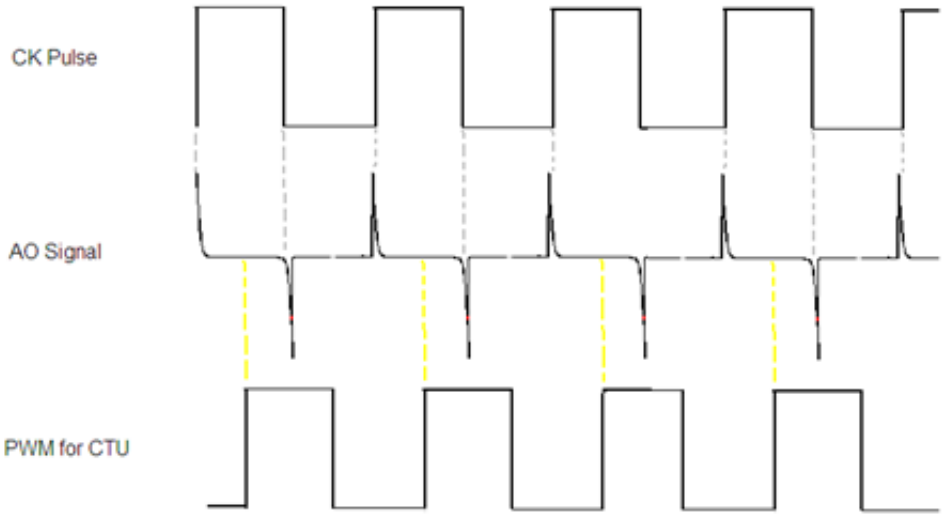

Figure 10. Processing with ADC and CTU the AO input signal

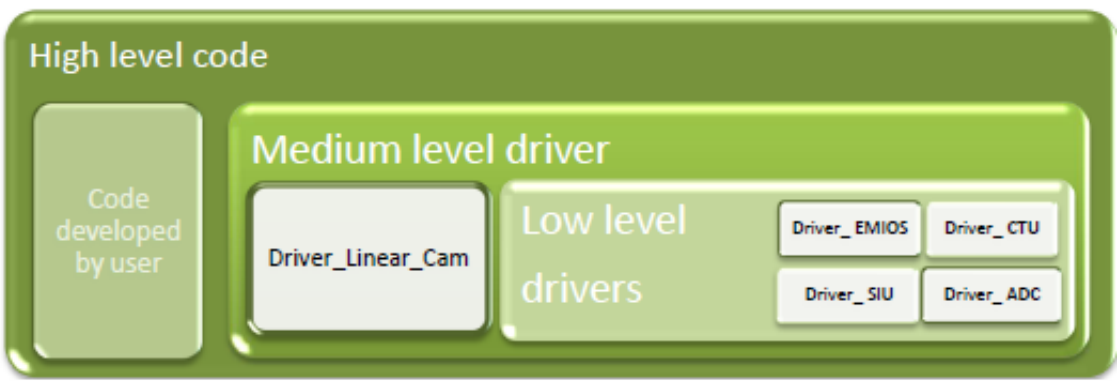

Figure 11. API implementation-version 2

According to Figure 15, the user has to call for functions from the *Driver_Linear_Cam.c* driver, which at the same time calls for functions from low level drivers that corresponds to functions from the *Driver_EMIOS.c*, *Driver_SIU.c* , *Driver_CTU.c* and *Driver_ADC.c* drivers. Then, to obtain the values of the line width and position of the input signal (detecting the line) for your high level algorithm, it is necessary to call the functions *u16_LineWidth()* and *u16_Center()* ; these are just return functions for the *u16LineWidth* and the *u16Center* variables.

Table 6. Advantage and disadvantage of processing with ADC and CTU

Advantage	Disadvantage
No external analog processing needed	If the user accelerates the frame rate then the ADC through-put to the MCU can be excessive
Never faces problems for calibrating	

Table 7. Driver_Linear_Cam: vfnInit_CamLin_Adc() function

Function	vfnInit_CamLin_Adc	Set configurations in ADC, CTU and EMIOS modules to read AO signal of the camera AO. Set global variables to 0.
Parameters		None
Return		Null
Functions from SIU	vfnInit_Emios_Input_Pad	Initialize a microcontroller pin as an input for an EMIOS channel.
	vfnInit_Emios_Output_Pad	Initialize a microcontroller pin as an output for an EMIOS channel.
	vfnInit_Adc_Pad	Initialize a microcontroller pin as an input for an ADC channel.
Functions from EMIOS	vfnInit_Emios_0_Opwm	Define EMIOS channel as positive OPWM with time base corresponding to the counter bus B,C,D,or E and establish its raising and falling edge with parameters A and B.
	vfnInit_Emios_0_Saic_Ctu	Define EMIOS channel as SAIC for CTU purpose with desired polarity and time base corresponding to the counter bus B,C,D, or E, allow channel freezing.
Functions from CTU	vfnTrigger_Emios_Adc	Configure event on an EMIOS channel to trig a conversion of a selected ADC channel.
Functions from ADC	vfnInit_Adc_Ctu	Initialize ADC in scan mode, configure ADC clock to 32 MHz, set selected ADC channel from a channel type as a cross triggering ADC channel.
Functions from ADC	PCR_EMIOS_0_tag	Pad configuration register required to configure SIU module. Defined in “Driver_MPC5604B”.
	PCR_ADC_tag	Pad configuration register required to configure SIU module. Defined in “Driver_MPC5604B”.
	ADC_tag_CHANNEL	ADC channel type, options are: precision, internal extended or external ADC channel. Defined in “Driver_MPC5604B”.

Table continues on the next page...

Table 7. Driver_Linear_Cam: vfnInit_CamLin_Adc() function (continued)

Definitions involved	ADC_CHANNEL_tag	Value of ADC channel for ADC module. Defined in “Driver_MPC5604B”.
	CTU_CHANNEL_tag	Value of ADC channel for CTU module. Defined in “Driver_MPC5604B”.
	CHAN_IN_TRIG	EMIOS channel declared as SAIC that will cause the cross triggering.
	CHAN_OU_TRIG	EMIOS channel declared as OPWM that will produce the out of phase clock pulse for the cross triggering.
	RISING_EDGE	Value of polarity to be captured by SA-IC. Defined in “Driver_MPC5604B”.
Variables involved	u8ScanDone	Global variable set to 0.
	u8i	Global variable set to 0.
	u16Center	Global variable set to 0.
	u16LineWidth	Global variable set to 0.

Table 8. Driver_Linear_Cam: u8Capture_Pixel_Values() function

Function	u8Capture_Pixel_Values	Set configurations in ADC, CTU and EMIOS modules to read AO signal of the camera AO. Set global variables to 0.
Parameters		None
Return	1	Line width and line center position values are finally captured and ready to be read.
	0	Line width and line center position values are not yet ready to be read.
Functions from EMIOS	EMIOS_0.CH[#].CCNTR.B.CCNTR	Register of actual counter value of selected EMIOS channel declared as modulus counter.
	vfnSet_Duty_Opwm	Establish duty cycle of an OPWM pulse where raising edge corresponds to value 0 of the counter and falling edge corresponds to the value selected by the user.

Table continues on the next page...

Table 8. Driver_Linear_Cam: u8Capture_Pixel_Values() function (continued)

Functions from ADC Definitions involved	u16Read_Adc	Checks for an ADC conversion to be complete, reads the value of the conversion, scale the read value in a range from 0 to Maximum Value (1023 for this case) and devolves the scaled value.
	CAM_ADC_CHAN	ADC channel used to convert camera pixel values from AO signal.
	CAM_CNT_CHAN	EMIOS channel used as Modulus Counter for SI signal.
	CHANNEL_CK	EMIOS channel used to generate CK signal for camera.
	CHAN_OU_TRIG	EMIOS channel declared as OPWM that will produce the out of phase clock pulse for the cross triggering.
Variables involved	u8ScanDone	Global variable used to specify if all pixels are captured and line width and line center position are calculated.
	u8i	Global variable used as counter for saving pixel conversion values in “u16Pixel” array.
	u16Center	Global variable used to save the value of position of the line center after calculations.
	u16LineWidth	Global variable used to save the value of the line width after calculations.
	u16Pixel	Global array use to save converted value of all pixels.
	u16CamCounter	Global variable used to save the actual value of the counter of the SI signal.
	u16cont	Local variable used as counter for an algorithm to obtain the minor value of the “u16Pixel” array.
	u16minval	Local value used to save the minor value of the “u16Pixel” array.
	u16minpos	Local value used to save the position of the minor value of the “u16Pixel” array.
	u16treshold	Local variable used in algorithm to calculate the position of the corners of the line in the “u16Pixel” array
	u16corner1	Local variable used to save the position of the beginning of the line in the “u16Pixel” array.
	u16corner2	Local variable used to save the position of the end of the line in the “u16Pixel” array.

6 Conclusion

As mentioned before, the first implemented method has two advantages and two disadvantages; from these, calibration is a big issue as light (used as reference) is a variable dependant on the environment. So, bad illumination, shadows, etc, will be factors that make this task difficult. On the other hand, if the user can fix this calibration issue, the advantages make this method the best option. The second method is more comfortable because external processing is not necessary and will not face calibration issues. The only problem is the use of the ADC which means throughput is added to the application; this can pose a problem depending on the rest of the tasks and sensing stuff the user may require for other processes. Based on the above implementations and after using the camera signal processing feature of MPC5604B MCU, the user can notice the huge capabilities this family of microcontrollers offers.

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