Cimarron Systems

CS36x DMSDK Camera Automatic Exposure

Application Note: AN103

December 1, 2012

Cimarron Systems, LLC

Table of Contents

Introduction 1
DMSDK Development Environment 1
RTP A/V Streaming Client/Server Application Overview 1
DMSDK Camera Automatic Exposure Algorithm 2
Automatic Exposure Algorithm Implementation 4
Additional Resources 6

List of Figures

Figure 1: DMSDK Development Context Diagram. 1
Figure 2: RTP A/V Streaming Client/Server Application Architecture. 2
Figure 3: Practical Dynamic Range of the Typical CMOS RGB Sensor. 2
Figure 4: Conceptual View of the CS36x DMSDK AEW Engine Statistics Gathering Process. 3
Figure 5: Initial Region Allocation for Automatic Exposure Algorithm. 4
Figure 6: Video Camera End-to-end Processing. 5

Document: AN-121201-1
Document Status: Released
Revision: 1.0

Introduction

The CS365-TI, CS368-TI, and CS368-LI Digital Media Software Development Kits (DMSDK) each implements an automatic exposure function that adjusts the camera's CMOS sensor RGB gain set point under environmental lighting conditions as seen through the lens. The Automatic Exposure (AE) Algorithm implemented contained within the DMSDK is designed to illustrate the use of the TMS320DM36x IPIPE Automatic Exposure/Automatic White Balance (AEW) Engine—for our discussion, simply the AEW Engine.

DMSDK Development Environment

Figure 1 shows a context diagram of the typical development environment in which the DMSDK operates, including: the TM320DM36x DVEVM running a number of the DMSDK components, a Ubuntu Linux Host computer, an audio/video source, and a video display/audio output device. Additionally, the LI-5M02 Camera Board—which is interfaced to the TMS320DM36x DVEVM—utilizes the Aptina MT9P031 5 Megapixel CMOS Digital Image Sensor, a manually adjustable CS mount lens, and an I^2C serial link connection to the DVEVM.[1]

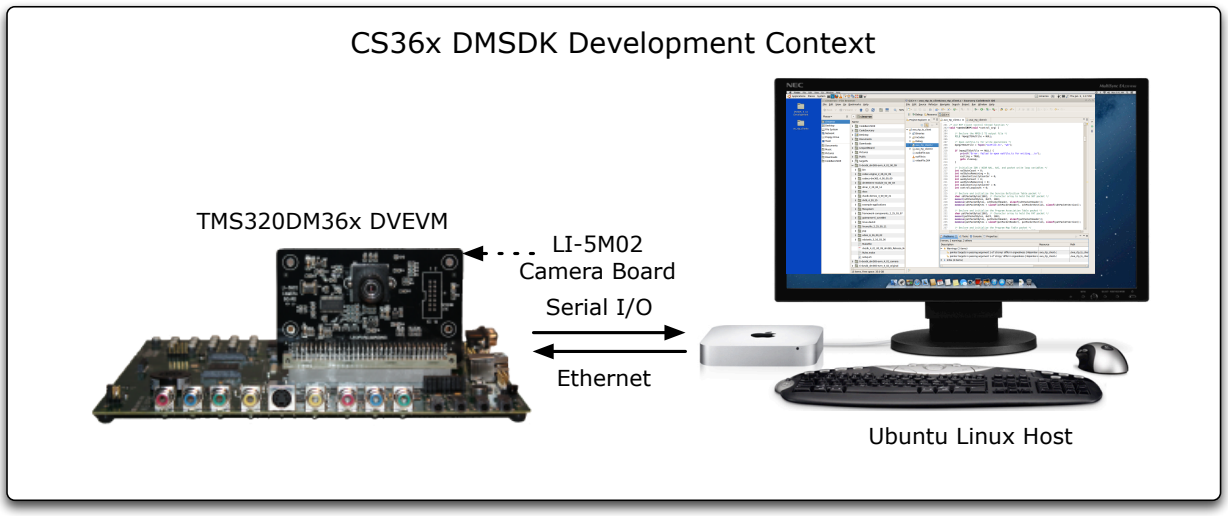

Figure 1: DMSDK Development Context Diagram.

RTP A/V Streaming Client/Server Application Overview

As illustrated in Figure 2, RTP A/V Streaming Server and RTP A/V Streaming Client Applications include: a RTCP Transmitter User Agent; an A/V Encoder set for encoding video to H.264 and audio to AAC-LC; and a RTP Transmitter. The RTP A/V Streaming Client component includes: a RTP Receiver; a RTCP Receiver User Agent; and a MPEG-2 Transport Stream (TS) Encoder that multiplexes the encoded H.264 video and AAC-LC audio into a transport stream with a switchable File Writer.[2]

During development, the CS36x DMSDK can select among the YP_rP_b Component, NTSC/PAL D1, S-Video, and LI-5M02 Camera Board; however, in this application note we are concerned with the camera board as the video source. Specifically, we will focus on the Image Sensor Interface (ISIF) as well as the Hardware 3A

---

1 Details described in this Application Note also apply to the CS368-LI DMSDK for the Leopard Board 368.

2 Both the RTP A/V Streaming Server and RTP A/V Streaming Client Applications implement H.264 video and AAC audio streaming in accordance with IETF RFC 3550 RTP: A Transport Protocol for Real-Time Applications, RFC 6184 RTP Payload for H.264 Video, RFC 3550 RTP: A Transport Protocol for Real-Time Applications, and RFC 3551 RTP Profile for Audio and Video Conferences with Minimal Control.

(H3A) Module with its ISIF statistics gathering capabilities—these interfaces and capabilities are described in detail in the Texas Instruments document *TMS320DM36x Digital Media Systems-on-Chip (DMSoC) Video Processing Front End (VPFE) User's Guide* (in order to learn how to implement Statistics Windows in the Automatic Exposure Algorithm, the reader will need to contact their local Texas Instruments FAE to obtain the NDA version of this document).

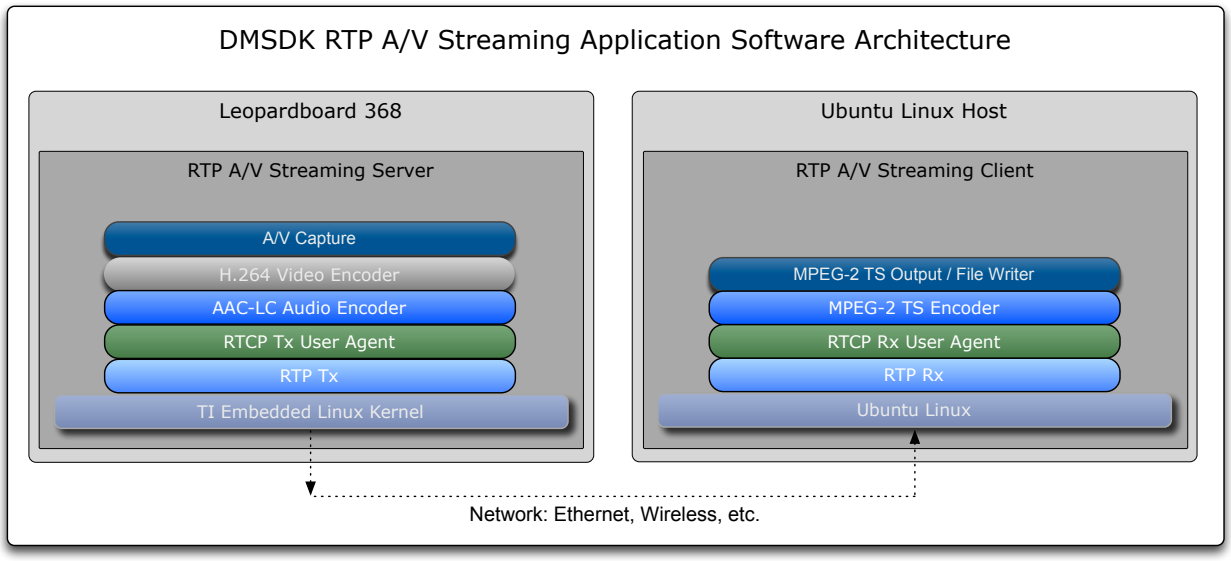

Figure 2: RTP A/V Streaming Client/Server Application Architecture.

DMSDK Camera Automatic Exposure Algorithm

While the brightness of a specific scene may vary as much as 160 dB, Figure 3 shows that the practical brightness range of the typical CMOS RGB sensor is on the order of 55 dB.

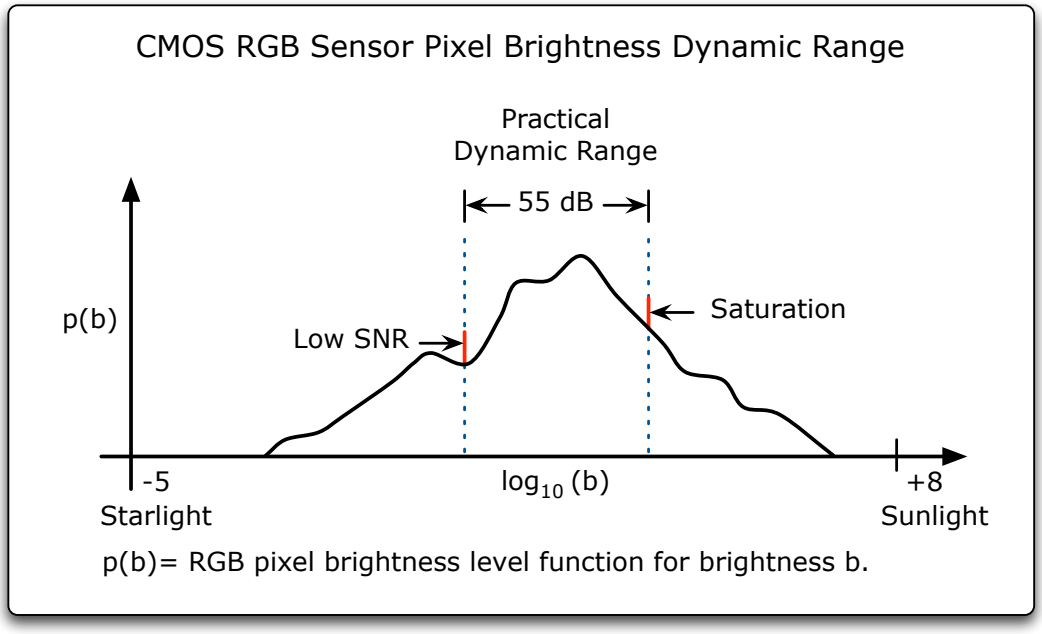

Figure 3: Practical Dynamic Range of the Typical CMOS RGB Sensor.

As a result, the design of a robust automatic exposure algorithm is extremely challenging since, at low brightness levels the sensor may be operating near its signal-to-noise ratio limit—subject to scene under exposure—while at high brightness levels the sensor to become saturated—portions of the scene are subject to overexposure (or "blooming"). This is where the computational capabilities of the DM36x AEW Engine become essential. For an excellent theoretical as well as practical treatment of the automatic exposure topic, please see *Camera Auto Exposure Control for VSLAM Applications*, by Michael Muehlebach, Swiss Federal Institute of Technology Zurich, 2010 (for a link to this document, please see the Additional Resources section below).

As illustrated in Figure 4, the DM36x VPFE H3A module's statistics gathering capability—the AEW Engine—does its work by positioning a "statistical window" within a selected area of the video sensor's viewport then performs certain selectable mathematical operations on the intensity values of the enclosed RGB pixels. For example, the DM36x AEW Engine is capable of operating on the RGB pixel intensities by computing: 1) the value of the sum-of-the-squares of the intensities; 2) the simple summation of the intensities; or 3) the minimum and maximum values of the intensities. In addition, the number of statistical windows can be increased and can be strategically placed within video sensor's viewport. Finally, the statistical window (or windows) can be moved around within the sensor's viewport dynamically and, within reasonable limits, on a frame-by-frame basis thereby allowing the engineer to develop camera automatic exposure algorithms that can range from simple yet effective to very complex so that it may adapt to extreme lighting environments.

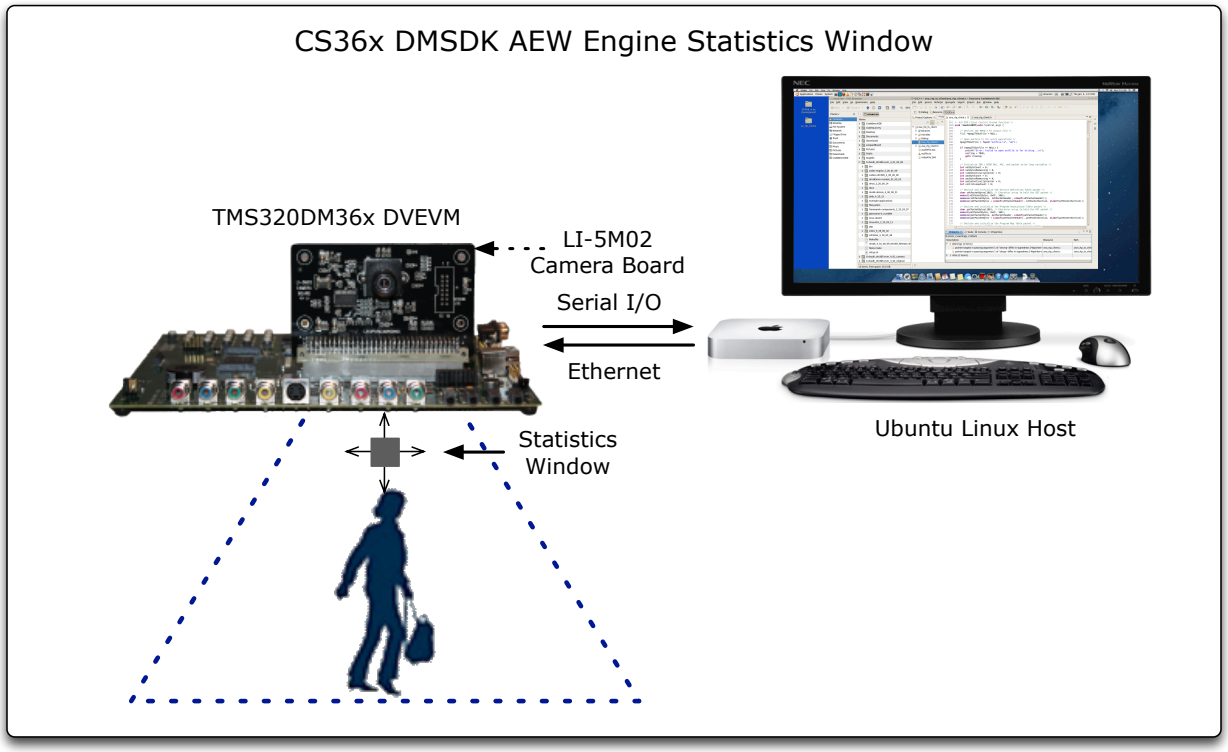

Figure 4: Conceptual View of the CS36x DMSDK AEW Engine Statistics Gathering Process.

If we assume that the object-of-interest is stationary, Figure 5 provides an example of how we might initially allocate regions of a CMOS RGB sensor's viewport so that the Automatic Exposure Algorithm may determine the proper RGB Gain setting. With this region allocation, the algorithm would proceed as follows:

1. Set up a single statistics window in each region that measures average pixel intensity then search each of the regions in order to confirm (or deny) that the object-of-interest occupies Regions 2 and 4 then save each region's window data.
2. Assuming that the search confirms the object's location, set up multiple statistics

windows—say four each—in Regions 2 and 4 that measures both average as well as minimum-maximum intensity values then save each region's windows data.

3. Using the statistics window values for Regions 1, 3, and 5 gathered in Step 1 as well as the statistics windows values for Regions 2 and 4 gathered in Step 2, compute the optimum RGB Gain value for the scene.

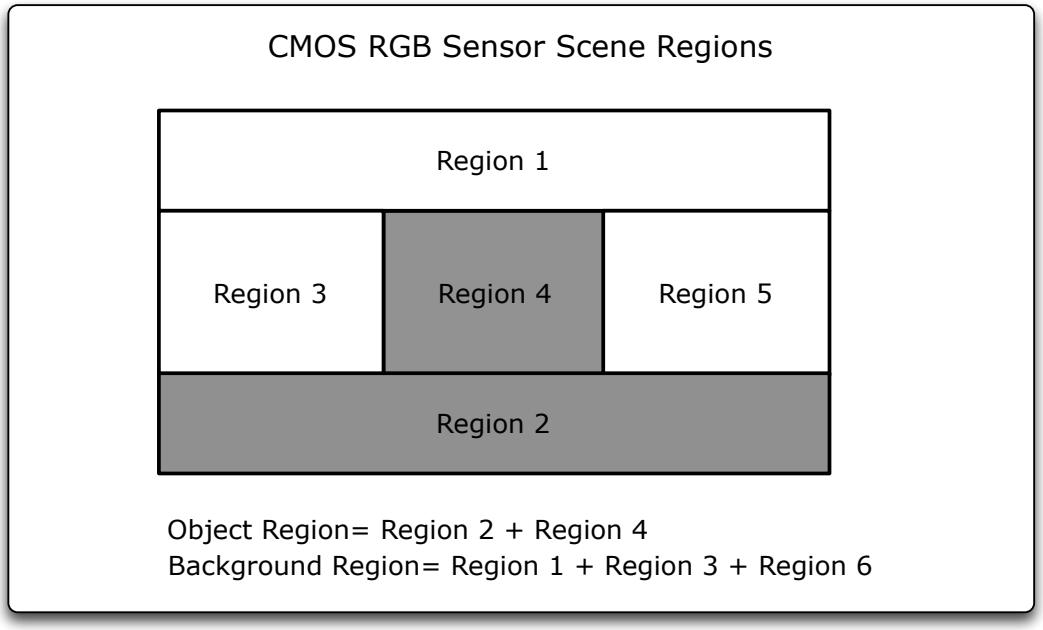

Figure 5: Initial Region Allocation for Automatic Exposure Algorithm.

In the next section, we will examine how the the CS36x DMSDK implements its Automatic Exposure Algorithm, recommendations for extending the algorithm, and suggests for further study by the reader.

Automatic Exposure Algorithm Implementation

In order to synchronize the viewport windows statistics gathering process, the Automatic Exposure Algorithm is implemented inside the 'A/V Capture Thread' of the RTP A/V Streaming Server. The Sequence Diagram shown in Figure 6 illustrations the entire process:

1. In the 'capture thread()':
   a) through the 'H3A/ISIF', the AEW Engine is initialized.
   b) within the thread, the 'Video Capture Interface' is initialized.
   c) through the 'H3A/ISIF', the Statistics Windows are initialized.
   d) through the 'DMAI/video thread()', the Video Codec is initialized.
   e) through the 'DMAI/video thread()', the Encode Buffers are initialized.
   f) within the thread, the 'Video Capture Interface' captures a Video Frame then passes it to the H.264 Video Encoder.
   g) through the 'H3A/ISIF', the Statistics Windows values are read and the RGB Gain is set.
   h) through the 'H3A/ISIF', new Statistics Windows locations are computed then set up.

i) the thread loops back to 1f) until the 'globalQuit' signal is received then the 'cleanup' process shuts down and cleans up the thread.

2. In the 'video thread()':
   a) initialize the thread, initialize the Video Capture Interface (DMAI), and initialize Video Code/Encoder Buffers (DMAI).
   b) encode captured Video Frame then pass H.264 Frame to the 'av_rtp_tx thread()'.
   c) the thread loops back to 2b) until the 'globalQuit' signal is received then the 'cleanup' process shuts down and cleans up the thread.
3. In the 'av_rtp_tx thread()':
   a) initialize the thread and sockets.
   b) within the thread, wait for a new H.264 Frame, receive/packetize the H.264 Frame, and send the RTP Video Packet.
   c) the thread loops back to 3b) until the 'globalQuit' signal is received then the 'cleanup' process shuts down and cleans up the thread.

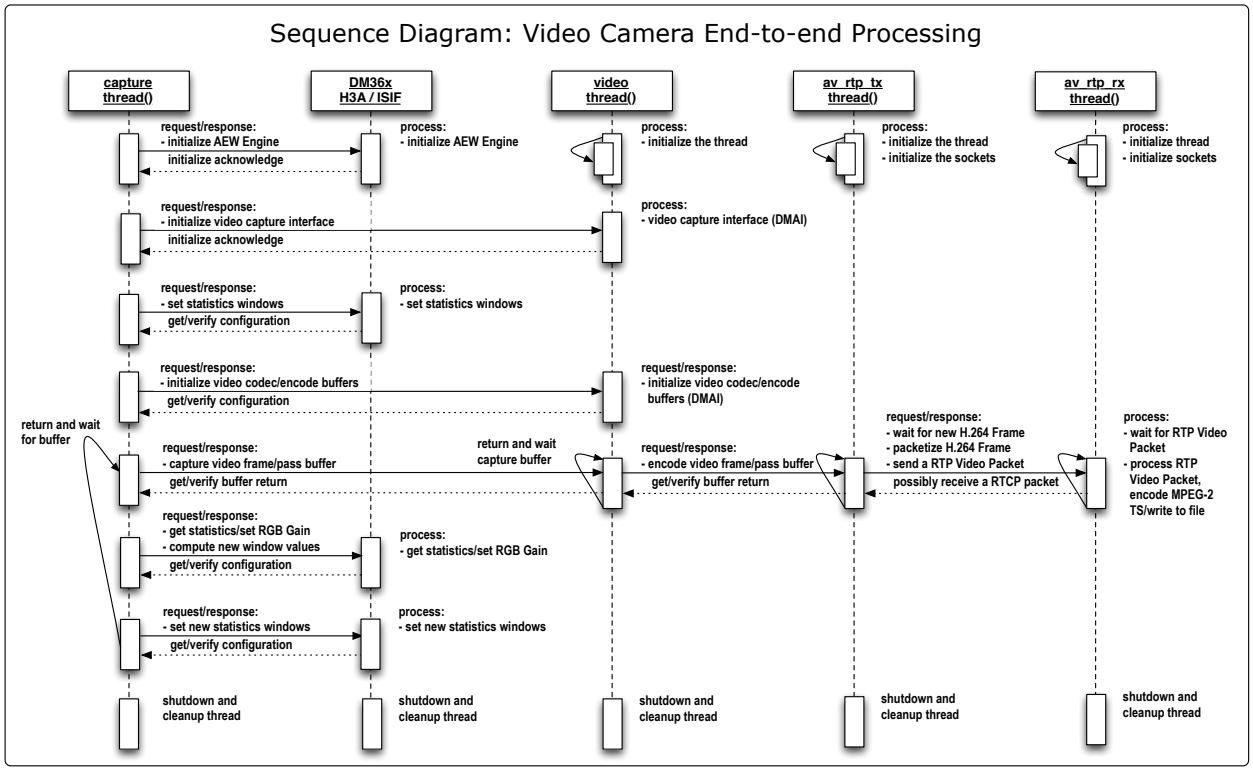

Figure 6: Video Camera End-to-end Processing.

4. In the 'av_rtp_rx thread()':
   a) initialize the thread and sockets.
   b) within the thread, wait for a new RTP Video Packet, receive/depacketize the RTP Packet into a H.264 Frame, encode the PES MPEG-2 TS, and, optional, write the video frame to an H.264 file.

c) the thread loops back to 4b) until the 'globalQuit' signal is received then the 'cleanup' process shuts down and cleans up the thread.

In this Application Note, we have presented an overview of the core elements of the CS36x DMSDK Automatic Exposure Algorithm, nevertheless, the larger issue surrounding the implementation of the camera viewport search algorithm has not been addressed in any detail. This is an exercise left to the reader!

Additional Resources

In addition to those already described, listed below are a number of resources that may be helpful:

1. Automatic Exposure Paper
2. Applied Image Processing eBook
3. Automatic Exposure Paper Digest

For more information regarding this and other Cimarron Systems products, please contact us using the contact information below.

Contact Information:

Cimarron Systems, LLC
Evergreen, Colorado
telephone: (303) 674-9207
email: info@cimarronsystems.com
www.cimarronsystems.com

Revision History:

Date	Version	Notes
12/1/2012	version 1.0	Initial version

