Camera Tamper Detection for ASC884xA/5xA

Doc Rev 0.0.1 Draft — 05 March 2013 January 2013 SH&E/EB

Document information

Info	Content
Document Type	Application Note for Camera Tamper Detection for ASC884xA/5xA
Author	Huzaifa Najmi
Keywords	IP Camera, Camper Tamper Detection
Abstract	

Revision history

Revision	Date	Description	Author
0.0.1	05 March 2013	First Version.	Huzaifa Najmi

Contents

1. Definitions

IVS : Intelligent Video Surveillance

SDK : Software Development Kit

2. Introduction

2.1 Purpose

This document describes usage of ASC884xA/5xA IVS Hardware accelerators to accomplish Camera Tamper Detection. It provides a sample implementation as an example for demonstrating the usage of such accelerators.

2.2 Scope

This document is intended to serve as a reference for usage of ASC884xA/5xA IVS Hardware accelerators useful for Camera Tampering Detection. It also refers to a sample implementation in the SDK.

IVS applications can be customized with more sophisticated algorithm for efficient Camera Tampering Detection making use of the IVS hardware accelerators.

2.3 General

Camera Tampering Scenarios:

- Spray Paint on the Camera Lens
- De-focusing : Changing the focus of the Camera Lens
- Blocking the Camera lens with an object say a Cloth, hand, cardboard, and so on.

3. Camera Tampering Detection

ASC884xA/5xA can robustly detect Camera Tampering by combining the below two methods.

3.1 Camera out of focus

ASC884xA/5xA can detect the tampering of camera like de-focusing. In case somebody changes the focus by manually adjusting the lens assembly and field of view is blurred ASC884xA/5xA can detect it with appropriate software.

ASC884xA/5xA has dedicated Hardware accelerators which can provide the focus value of the scene to the application. It can even be programmed to record this information at regular intervals like one per 10 frames. The application software can access this information and if the focus value varies above a threshold it can generate an alarm.

In addition, ASC884xA/5xA also provides focus values for 7x7 grid . Advanced algorithms could make use of these values.

3.2 Camera Scene Change

ASC884xA/5xA can detect blocking the camera view with tape, cloth, form or paint. Like the de-focus application mentioned above, ASC884xA/5xA can provide the sum of red, blue and green values of the entire image. These values can be used by application software to detect sudden variations in color and intensity of the overall scene and also detect a scene change. When somebody blocks the view of camera there will be a sudden variation of color and intensity. When such variation is above threshold, it can report an alarm.

In addition, ASC884xA/5xA also provides R, G , B Sum and histogram values for 16x15 grid . Advanced algorithms could make use of these values.

4. Example Code:

NXP ASC884xA/5xA SDK provides a reference example application to demonstrate its capabilities for Camera Tamper Detection with a simple algorithm. The application can be further customized to allow for more advanced algorithms to be implemented based on the information provided by ASC884xA/5xA.

The Tamper Detect algorithm is as follows:

Step 1: Compare Current Vs Previous Focus Value

Check if the focal value is reduced beyond a predefined and configurable threshold (“FOCUS_DIFF_THRES_HIGH” & “FOCUS_DIFF_THRES_LOW”) compared to the previous frame focus value. This comparison is done after skipping “NO_OF_FRAMES” number of consecutive frames.

Step 2. Compare Current Vs Average Focus Value

If Step 1 is true, check if the focus value is reduced beyond a predefined and configurable threshold (“FOCUS_DIFF_THRES_AVG_HIGH” & “FOCUS_DIFF_THRES_AVG_LOW”) compared to the **average** focus value.

Step 3: Compare Current R/G/B Sum with Average R/G/B Sum

If Step 1 and Step 2 both are true, check if the Red Sum, Green Sum and Blue Sum has reduced beyond a predefined and configurable threshold (“FOCUS_DIFF_THRES_AVG_HIGH” & “FOCUS_DIFF_THRES_AVG_LOW”) compared to the average Red, Green and Blue Sum values. This check can also be modified to compare the Sum of Red, Green and Blue Sums (RSum+GSum+BSum) with the average sum of Red Sum, Green Sum and Blue Sum.

The following parameters can be customized:

1. Focus Value Difference threshold
2. Focus Value Difference threshold for average
3. Number of frames to skip for every focus value reading
4. Number of frames to use for computing the averages
5. Total number of frames for the application to run.

The reference application viz.: TamperDetect works as a slave to the IPCam Reference application, Kilrogg. It thus requires that Kilrogg is already running on the target. It requires

the sensor configuration file to be passed as an argument. More details on Sensor configuration file are available in Sec. 4.3 of this document.

4.1 Data Structure for Focus Values and R, G, B Sum:

4.1.1 TVideoCapInitOptions

This structure defines all parameters for the Video Capture library during initialization and is used by the VideoCap_Initial function.

```
typedef struct video_cap_state
{
   /* --- Other fields here------*/
   BYTE *pbyStatAEWBBufUsrBaseAddr;
   BYTE *pbyStatHistoBufUsrBaseAddr;
   BYTE *pbyStatFocusBufUsrBaseAddr;
   /* --- Other fields here------*/

} TVideoCapState;
```

Field	Description
pbyStatAEWBBufUsrBaseAddr	Start address of 16*15 grid window R G B sum statistics. 4 bytes for 1 grid window statistic value. First 960 bytes for R , and second 960 bytes for G, last 960 bytes for B.
pbyStatHistoBufUsrBaseAddr	Start address of R G B histogram statistics. 4 bytes for 1 bin value. RGB have 64 bins separately. First 256 bytes for R , and second 256 bytes for G, last 256 bytes for B.
pbyStatFocusBufUsrBaseAddr	Start address of 7*7 grid window focus statistic. 8 bytes for 1 grid window statistic value.

NOTE:

Current reference application for Tamper detection :

- Does Not use 16*15 grid values of R,G, B sum statistics
- Does Not use Histogram Values
- Does Not use 7*7 Focus statistics.

However, customer can build advanced algorithm which may like to use these values.

4.1.2 TVideoCapState

This structure sends and receives the information of current frame through the VideoCap_GetBuf function.

```
typedef struct video_cap_state
{
   /* --- Other fields here------*/

        DWORD dwAFFocusValueHigh;
        DWORD dwAFFocusValueLow;
```

```
    QWORD qwAWBRedSum;
    QWORD qwAWBGreenSum;
    QWORD qwAWBBlueSum;

} TVideoCapState;
```

Field	Description
dwAFFocusValueHigh	32 bit MSB of AF focus value information.
dwAFFocusValueLow	32 bit LSB of AF focus value information. Combined with dwAFFocusValueHigh to get the focus measure statistics which could be used for focus lens control. The higher the focus measure statistics is, the clearer the image is.
qwAWBRedSum	AWB Red summation information.
qwAWBGreenSum	AWB Green summation information.
qwAWBBlueSum	AWB Blue summation information.

4.2 Reference Example Code:

Path: *NXP_ASC88xx_SDK.6.x/IPCAM_Reference/Application/Kilrogg_r45894_IPCam/external/TamperDetect*

File: TamperDetect.c

Key areas of the code are reproduced below:

```
#include "TamperDetect.h"

/*==============================     Definitions  ================================*/

/*============    Modify the Previous Focus Threshold here  =================*/

#define FOCUS_DIFF_THRES_HIGH    (0x0)

#define FOCUS_DIFF_THRES_LOW     (0x1000000)

/*==============   Modify the Focus Average Thresholds here  ==============*/

#define FOCUS_DIFF_THRES_AVG_HIGH    (0x0)

#define FOCUS_DIFF_THRES_AVG_LOW    (0x40000000)

/*=============   No of Previous frames to save focus average and RGB Sum ==========*/

#define NO_OF_FRAMES     (10U)

/*===================     Modify the Frame Skip Period here ================================*/

#define FRAME_SKIP_PERIOD (10U)

/*===================    Modify the R/G/B Sum Thresholds here  ============================*/

#define R_SUM_THRESHOLD                         (0x1000000)

#define G_SUM_THRESHOLD                         (0x1000000)

#define B_SUM_THRESHOLD                         (0x1000000)


#define MSB_32_SHIFT                                (32U)
```

```
#define LSB_32_MASK                                  (0xffffffff)

/*==========  Macro Definitions  =================*/

#define QWORD_2_DWORD(qwValue, dwHighValue, dwLowValue)                  \
  {                                                                      \
       dwHighValue = (DWORD ) ((qwValue ) >>MSB_32_SHIFT);               \
       dwLowValue  = (DWORD ) ((qwValue) & LSB_32_MASK);                 \
  }
#define DWORD_2_QWORD(qwValue, dwHighValue, dwLowValue)                  \
 {                                                                       \
     qwValue= ((((QWORD)(dwHighValue)) << MSB_32_SHIFT) |(((QWORD)(dwLowValue)) &LSB_32_MASK )); \
}
/* ================================================================== */
int main(int argc, char* argv[])
{
/* ----- Other code -------------------------*/
tVideoCapState.dwFrameCount =0;
dwInitalFrameCount = 0;
while ( tVideoCapState.dwFrameCount   <= (MaxNoOfFrames + dwInitalFrameCount)  )
{
     if (VideoCap_TamperDetect(hVideoCapObject, &tVideoCapState, &bIsTamperDetect,
(PDWORD)tVideoCapInitOptions.pbyStatFocusBufUsrBaseAddr,(PBYTE)tVideoCapInitOptions.pbyStatHistoBufUsrBaseAddr) != S_OK)
   {  DBPRINT_L1_0("VideoCap_TamperDetect:  --------------- Error getting video capture buffer-------------\n\n");   }
  else if (bIsTamperDetect == TRUE)
   {  DBPRINT_L1_0("\nVideoCap_TamperDetect:  ---------------    Camera Tampering DETECTED            !!!!!!!!!!!!!!!!!!!!!!!!!!!!    -------\n\n");
   } else
   { DBPRINT_L3_0  ("VideoCap_TamperDetect:  ------------       Camera Tampering      N O T   DETECTED              ----------\n\n");    }

 if (dwInitalFrameCount == 0)   {
       dwInitalFrameCount = tVideoCapState.dwFrameCount;
    }
  }
}

SCODE  VideoCap_TamperDetect(
                             HANDLEh Object,
                             TVideoCapState *ptState,
                             BOOL *pbIsTamperDetect,/* Output Variable */
```

```
                            PDWORD pdwStatFocusBuf,
                            PBYTE  pbyStatHistoBuf
                            )
{
  static DWORD dwOOF_PrevFramesAvg_High =0; static DWORD dwOOF_PrevFramesAvg_Low =0;
  static DWORD dwOOF_PrevFrames_High[NO_OF_FRAMES]={0};  static DWORD dwOOF_PrevFrames_Low[NO_OF_FRAMES]={0};
  static QWORD qwTD_R_Avg =0;static QWORD qwTD_G_Avg =0;static QWORD qwTD_B_Avg =0;
  static QWORD qwTD_R_Sum[NO_OF_FRAMES]={0};static QWORD qwTD_G_Sum[NO_OF_FRAMES]={0}; static QWORD qwTD_B_Sum[NO_OF_FRAMES]={0};
  static DWORD dwArrayIndex=0; static DWORD dwFrameNum=     0; static DWORD dwPrevFrameCount  =0;

  *pbIsTamperDetect =FALSE; //Initialise
  dwPrevFrameCount = 0xFFFFFFFF;

  VideoCap_Sleep(hObject); // Sleep before the check so that we dont keep printing /wasting CPU while we need to skip frames.
  if ( (dwFrameNum % FRAME_SKIP_PERIOD) == 0)
  {    /*-------------- Get the current frame ----------*/
         scStatus = VideoCap_GetBuf(hObject, ptState);
         {        //Check for errors   }

  // To reach here, scStatus must be S_OK and a new frame is captured.
  /*----------------------------  Update the Average   ------------------------*/
  {
  static BOOL dwArrayFull = FALSE;
  if(dwArrayFull)
  {
      dwOOF_PrevFramesAvg_High += (ptState->dwAFFocusValueHigh/NO_OF_FRAMES )   -  (dwOOF_PrevFrames_High[dwArrayIndex] /NO_OF_FRAMES) ;
      dwOOF_PrevFramesAvg_Low += (ptState->dwAFFocusValueLow /NO_OF_FRAMES )   - (dwOOF_PrevFrames_Low[dwArrayIndex] /NO_OF_FRAMES );

      qwTD_R_Avg+= ( ptState->qwAWBRedSum      /  NO_OF_FRAMES )       - (qwTD_R_Sum[dwArrayIndex]/NO_OF_FRAMES);
      qwTD_G_Avg+=( ptState->qwAWBGreenSum  /   NO_OF_FRAMES )      -  (qwTD_G_Sum[dwArrayIndex]/NO_OF_FRAMES);
      qwTD_B_Avg+=( ptState->qwAWBBlueSum     /   NO_OF_FRAMES )      -  (qwTD_B_Sum[dwArrayIndex]/NO_OF_FRAMES);
  }
  else
  {
      DWORD dwLoopCount =0;

      dwOOF_PrevFramesAvg_High =dwOOF_PrevFramesAvg_Low =0;
```

```
    qwTD_R_Avg = qwTD_G_Avg = qwTD_B_Avg =0;


    dwOOF_PrevFrames_High[dwArrayIndex ]=ptState->dwAFFocusValueHigh;
    dwOOF_PrevFrames_Low[dwArrayIndex ]=ptState->dwAFFocusValueLow;


    qwTD_R_Sum[dwArrayIndex] = ptState->qwAWBRedSum;
    qwTD_G_Sum[dwArrayIndex] = ptState->qwAWBGreenSum;
    qwTD_B_Sum[dwArrayIndex] = ptState->qwAWBBlueSum;


    for(dwLoopCount =0;   dwLoopCount  < (dwArrayIndex+1); dwLoopCount++)
    {
        dwOOF_PrevFramesAvg_High+=   (dwOOF_PrevFrames_High[dwLoopCount ]  /  (dwArrayIndex+1));
        dwOOF_PrevFramesAvg_Low+=   (dwOOF_PrevFrames_Low[dwLoopCount ]   /   (dwArrayIndex+1));


        qwTD_R_Avg +=  (qwTD_R_Sum[dwLoopCount ]  /  (dwArrayIndex+1));
        qwTD_G_Avg +=  (qwTD_G_Sum[dwLoopCount ]  /  (dwArrayIndex+1));
        qwTD_B_Avg +=  (qwTD_B_Sum[dwLoopCount ]  /  (dwArrayIndex+1));
    }
}


if (dwArrayIndex == (NO_OF_FRAMES-1) )  // dwArrayIndex  points to the current index
{
    dwArrayFull =TRUE;
}
}


/* check if current Focus Value is reduced beyond threshold wrt Prev stored focus value */
if( IsFocusValueReduced(ptState, dwOOF_PrevFrames_High[dwArrayIndex], dwOOF_PrevFrames_Low[dwArrayIndex], FOCUS_DIFF_THRES_HIGH,
                        FOCUS_DIFF_THRES_LOW))
{
    DBPRINT_L2_0("\n-----    Focus Value reduced beyond threshold:              Current   Vs    PREVIOUS      ---------\n\n");
    /* check if current Focus Value is reduced beyond threshold wrt Prev stored Average focus value */
    if( IsFocusValueReduced(ptState, dwOOF_PrevFramesAvg_High, dwOOF_PrevFramesAvg_Low, FOCUS_DIFF_THRES_AVG_HIGH,
                            FOCUS_DIFF_THRES_AVG_LOW ))
    {
        DBPRINT_L1_0("\n-----    Focus Value reduced beyond threshold:              Current   Vs   AVERAGE      ---------\n\n");
        /*Check if scene has changed based on information from R/G/B Sum */
```

```
        if( IsSceneChanged(ptState,qwTD_R_Avg, R_SUM_THRESHOLD, qwTD_G_Avg, G_SUM_THRESHOLD, qwTD_B_Avg, B_SUM_THRESHOLD ) == TRUE)
        {
                *pbIsTamperDetect =TRUE;
                 DBPRINT_L4_0("-------------------    SCENE  Change   D E T E C T E D  !!! -------------------\n");
        } else {
                *pbIsTamperDetect =FALSE;
                 DBPRINT_L2_0("-------------------    SCENE  Change   N O T  Detected!!! -------------------\n");
        }
    }
}
/*---------------------- Updates for next iterations----------------------*/
/* Update Current Focus Value to Array for next iterations of average calculations *//* This will be done twice when array is not
completely full*/
dwOOF_PrevFrames_High[dwArrayIndex] = ptState->dwAFFocusValueHigh;
dwOOF_PrevFrames_Low[dwArrayIndex] = ptState->dwAFFocusValueLow;
qwTD_R_Sum[dwArrayIndex] = ptState->qwAWBRedSum;
qwTD_G_Sum[dwArrayIndex] = ptState->qwAWBGreenSum;
qwTD_B_Sum[dwArrayIndex] = ptState->qwAWBBlueSum;

  /* Increment Array Index for next iteration*/
  dwArrayIndex = (dwArrayIndex+1) % NO_OF_FRAMES; // Since its a circular array index, indexed from 0
  /* Increment FrameCount */
   dwFrameNum++;
   }else {  //Skip Tamper Detection calculation for this frame dwFrameNum++; }
}

BOOL
IsFocusValueReduced(
                TVideoCapState *ptState,
                DWORD dwOOF_Prev_High,
                DWORD dwOOF_Prev_Low,
                DWORD dwFocus_Diff_Thres_High,
                DWORD dwFocus_Diff_Thres_Low
                )
{
       BOOL bIsFocusValueReduced =FALSE;
       QWORD qwOOF_Prev, qwFocus_Diff_Thres, qwAFFocusValue;
```

```
        DWORD_2_QWORD(qwOOF_Prev,              dwOOF_Prev_High,                 dwOOF_Prev_Low)
        DWORD_2_QWORD(qwFocus_Diff_Thres,    dwFocus_Diff_Thres_High,     dwFocus_Diff_Thres_Low)
        DWORD_2_QWORD(qwAFFocusValue,         ptState->dwAFFocusValueHigh,  ptState->dwAFFocusValueLow)

        if( qwOOF_Prev > qwFocus_Diff_Thres) // make sure below subtraction does not give negative result
        {
            if(qwAFFocusValue < (qwOOF_Prev -qwFocus_Diff_Thres) ) // subtracting here to avoid an MSB overflow during addition
            {
                bIsFocusValueReduced = TRUE;
                DBPRINT_L2_1("Low has reduced beyond threshold    bIsFocusValueReduced =%d\n",bIsFocusValueReduced);
            }
            else //Focus Value has not reduced beyond LOW threshold
            {
                bIsFocusValueReduced  = FALSE;
                DBPRINT_L3_1("Not reduced beyond threshold    bIsFocusValueReduced =%d\n",bIsFocusValueReduced);
           }
        }
        else
        {
            // Since qwOOF_Prev is smalleer than threshold, the focus value has obviously not reduced beyond the threshold values.
            bIsFocusValueReduced  = FALSE;
        }
  return bIsFocusValueReduced ;
}


BOOL IsSceneChanged(
                    TVideoCapState *ptState,
                    QWORD qw_R_Avg ,
                    QWORD qw_R_Thres ,
                    QWORD qw_G_Avg ,
                    QWORD qw_G_Thres ,
                    QWORD qw_B_Avg ,
                    QWORD qw_B_Thres
                    )
{
```

```
    BOOL    bIsSceneChanged =   FALSE;
    BOOL    bIs_R_Changed =   FALSE;
       BOOL    bIs_G_Changed=   FALSE;
       BOOL    bIs_B_Changed =   FALSE;

    if ( qw_R_Avg  > (ptState->qwAWBRedSum + qw_R_Thres ) )   /* Focus Value High has reduced beyond the High threshold. No need
    to check the Low value*/
    {               bIs_R_Changed  = TRUE;          }
    if ( qw_G_Avg  > (ptState->qwAWBGreenSum + qw_G_Thres ) )   /* Focus Value High has reduced beyond the High threshold. No need
    to check the Low value*/
    {               bIs_G_Changed  = TRUE;          }

    if ( qw_B_Avg  > (ptState->qwAWBBlueSum + qw_B_Thres ) )   /* Focus Value High has reduced beyond the High threshold. No need
    to check the Low value*/
    {               bIs_B_Changed  = TRUE;          }

/* A sample and simple logic for scene changed is introduced below. Feel free to add your own logic here. */
       if (  (bIs_R_Changed ==TRUE) && (bIs_G_Changed ==TRUE) && (bIs_B_Changed ==TRUE) )
       {               bIsSceneChanged = TRUE;          }
}
```

4.3 Sensor Configuration File

Path: *NXP_ASC88xx_SDK.6.x/IPCAM_Reference/Application/Kilrogg_r45894_IPCam/apps/venc/app/config_files/sensor_config*

The sensor configuration file needs to be modified as explained below:

Line 123:

0 // Focus statistic enable (0: disable, 1: enable)

➔ change to ➔

1// Focus statistic enable (0: disable, 1: enable)

Line 126:

10 // Focus statistic frame interval (1~15)

=> change appropriately to skip focus value calculation by ASC884xA/5xA chip

5. Acceptance Reviews and Approvals

Table 1: Accepters'

Name	Role	Location
Jaijith Radhakrishnan	System Architect	NXP Bangalore
Ashirwad Gujarathi	Design In Manager	NXP Bangalore

Table 2: Approvers

Name	Role	Location	Date	Signature (if required)
Jaijith Radhakrishnan	System Architect	Bangalore	<YYYYMMDD>	
Ashirwad Gujarathi	Design In Manager	Bangalore		
Vijayanand Jayaraman	SW Application Engineer	Bangalore		

6. Document Management

6.1 Referenced documents

Table 3: Referenced documents

Sr. no.	Doc Title	Version	Author	Issue Date
1	NXP_IVS_for_ASC884x_5x_v0.0.2.docx	0.0.2	Jaijith Radhakrishnan	17 January 2013
2	ASC8848A_49A_50A_51A_52A-v1.1.pdf	1.0	--	9 November2012
3	UM - Video-Capture_Library.pdf	1.36.0	--	19 December 2012