Ambarella

Ambarella System Software

API Specification

SDK6 API ADAS

Release : 1.1

Status : Release to Customer.

Last Update : August 6, 2015

Distributed For : Ambarella SDK

Ambarella Inc
3101 Jay Street
Ste.110
Santa Clara, CA 95054, USA
Phone: +1.408.734.8888
Fax: +1.408.734.0788

Ambarella Taiwan Ltd.
C1, 1F, No.1, Li-Hsin 1st Road,
Hsin-Chu Science-Based Park
Hsin-Chu City 30078, Taiwan ROC
+886-3-666-8828

Revision History

Date	Version	Descriptions
2015/5/19	1.0	Initial draft.
2015/7/23	1.1	Added Sections 2.24 – 2.33 (Motion Detection APIs).

Table of Contents

1.Introduction....1
1.1.Overview....1
2.List of APIs....2
2.1.Amba_Adas_Init....3
2.1.1.Amba_Ada_Init > AMP_ENC_YUV_INFO_s....3
2.2. Amba_Adas_Proc....4
2.2.1.Amba_Adas_Proc > AMBA_ADAS_AUX_DATA_s....4
2.2.2.Amba_Adas_Proc > AMBA_ADAS_GPS_INFO_s....4
2.2.3.Amba_Adas_Proc > AMBA_ADAS_DASHBOARD_INFO_s....4
2.2.4.Amba_Adas_Proc > AMBA_ADAS_OUTPUTEVENT_s....5
2.2.5.Amba_Adas_Proc > AMBA_ADAS_LDW_OUTPUTEVENT_s....5
2.2.6.Amba_Adas_Proc > AMBA_ADAS_DEPARTUREDIRECTION....5
2.2.7.Amba_Adas_Proc >AMBA_ADAS_LANEMARKTYPE....5
2.2.8.Amba_Adas_Proc > AMBA_ADAS_LINETYPE....5
2.2.9.Amba_Adas_Proc > AMBA_ADAS_LINESOLIDITY....5
2.2.10.Amba_Adas_Proc > AMBA_ADAS_LINECOLOR....6
2.2.11.Amba_Adas_Proc > AMBA_ADAS_FCW_OUTPUTEVENT_s....6
2.3. Amba_Adas_SetSceneParams....7
2.3.1.Amba_Adas_SetSceneParams > AMBA_ADAS_SCENE_PARAMS_s....8
2.3.2.Amba_Adas_SetSceneParams > AMBA_ADAS_CAMERA_MOUNT_HEIGHT_e....9
2.4. Amba_Adas_GetSceneParams....10
2.5.Amba_Adas_GetSceneStatus....11
2.5.1.Amba_Adas_GetSceneStatus > AMBA_ADAS_SCENE_STATUS_s....11
2.6. Amba_Adas_SetLdwParams....13
2.6.1.Amba_Adas_SetLdwParams > AMBA_ADAS_LDW_PARAMS_s....13
2.6.2.Amba_Adas_SetLdwParams > AMBA_ADAS_SENSITIVITY_LEVEL_e....13
2.7. Amba_Adas_GetLdwParams....14
2.8.Amba_Adas_GetLdwOutput....15
2.8.1.Amba_Adas_GetLdwOutput > AMVA_ADAS_LDW_OUTPUT_s....15
2.8.2.Amba_Adas_GetLdwOutput > AMBA_ADAS_RoadMarkLine_s....15
2.8.3.Amba_Adas_GetLdwOutput > AMBA_VA_POINT_s....15
2.9.Amba_Adas_SetFcwParams....16
2.9.1.Amba_Adas_SetFcwParams > AMBA_ADAS_FCW_PARAMS_s....16
2.10.Amba_Adas_GetFcwParams....17
2.11.Amba_Adas_GetFcwOutput....18
2.11.1.Amba_Adas_GetFcwOutput > AMBA_ADAS_FCW_OUTPUT_s....18
2.11.2.Amba_Adas_GetFcwOutput > AMBA_ADAA_FCW_WARNING_LEVEL_e....18
2.12.Amba_AdasLLWS_Proc....20
2.12.1.Amba_AdasLLWS_Proc > AMP_LLWS_PAR_t....21
2.13.Amba_AdasLLWS_Init....22
2.14.Amba_AdasLLWS_Deinit....23
2.15.Amba_AdasLLWS_SetCfg....24
2.15.1.Amba_AdasLLWS_SetCfg > AMP_LLWS_CFG_t....24
2.16.Amba_AdasLLWS_GetDefCfg....25
2.17.Amba_AdasLLWS_GetPar....26
2.18.Amba_AdasFCMD_Proc....27
2.18.1.Amba_AdasFCMD_Proc > AMP_FCMD_PAR_t....28
2.19.Amba_AdasFCMD_Init....29
2.20.Amba_AdasFCMD_Deinit....30
2.21.Amba_AdasFCMD_SetCfg....31
2.21.1.Amba_AdasLLWS_SetCfg > AMP_FCMD_CFG_t....31
2.22.Amba_AdasFCMD_GetDef_Cfg_Par....32
2.23.Amba_AdasFCMD_GetPar....33
2.24.Amba_AdasMDY_Proc....34
2.24.1.Amba_AdasMDY_Proc > AMBA_MDY_PAR_t....35
2.24.2.Amba_AdasMDY_Proc > AMBA_MDY_METHOD_e....35
2.24.3.Amba_AdasMDY_Proc > AMBA_MD_ROI_STATUS_t....35
2.24.4.Amba_AdasMDY_Proc > AMBA_MOTION_FLAGS_e....35

2.24.5.Amba_AdasMDY_Proc > AMBA_MOTION_EVENT_e....35
2.25.Amba_AdasMDY_Init....36
2.26.Amba_AdasMDY_Deinit....37
2.27.Amba_AdasMDY_GetDefCfg....38
2.27.1.Amba_AdasMDY_GetDefCfg > AMBA_MDY_CFG_t....39
2.27.2.Amba_AdasMDY_GetDefCfg > AMBA_MDY_ROI_DATA_t....39
2.28.Amba_AdasMDY_SetCfg....40
2.29.Amba_AdasMDAE_Proc....41
2.29.1.Amba_AdasMDAE_Proc > AMBA_MDAE_PAR_t....42
2.29.2.Amba_AdasMDAE_Proc > AMBA_MDAE_METHOD_e....42
2.30.Amba_AdasMDAE_Init....43
2.31.Amba_AdasMDAE_Deinit....44
2.32.Amba_AdasMDAE_GetDefCfg....45
2.32.1.Amba_AdasMDAE_GetDefCfg > AMBA_MDAE_CFG_t....45
2.32.2.Amba_AdasMDAE_GetDefCfg > AMBA_MDAE_ROI_DATA_t....46
2.33.Amba_AdasMDAE_SetCfg....47
3.Quick Start Guide with Example....48
3.1.Example: Quick Start ADAS Steps....48
3.2.Example: Quick Start FCMD Steps....50
3.3.Example: Quick Start LLWS Steps....51
3.4.Example: Print Calibration Levels in Manual Mode....52

1. Introduction

Advanced Driver Assistance Systems (ADAS), which is one of the fastest-growing segments in the automotive industry, help drivers in the driving process and are used to develop automated/enhanced vehicle systems that aid in driving and also help to avoid collisions and accidents.

1.1. Overview

Ambarella's Advanced Driver Assistance System (ADAS) includes four features:

- Lane Departure Warning System (LDWS)
- Front Collision Warning System (FCWS)
- Front Car Moving Detection (FCMD)
- Low Light Warning System (LLWS)

These features provide immense value in safety and driving.

This document provides the Ambarella Advanced Driver Assistance System (Amba ADAS) APIs for the A9, A9S, and A12 product lines.

2. List of APIs

- Amba_Adas_Init
- Amba_Adas_Proc
- Amba_Adas_SetSceneParams
- Amba_Adas_GetSceneParams
- Amba_Adas_GetSceneStatus
- Amba_Adas_SetLdwParams
- Amba_Adas_GetLdwParams
- Amba_Adas_GetLdwOutput
- Amba_Adas_SetFcwParams
- Amba_Adas_GetFcwParams
- Amba_Adas_GetFcwOutput
- Amba_AdasLLWS_Proc
- Amba_AdasLLWS_Init
- Amba_AdasLLWS_Deinit
- Amba_AdasLLWS_SetCfg
- Amba_AdasLLWS_GetDefCfg
- Amba_AdasLLWS_GetPar
- Amba_AdasFCMD_Proc
- Amba_AdasFCMD_Init
- Amba_AdasFCMD_Deinit
- Amba_AdasFCMD_SetCfg
- Amba_AdasFCMD_GetDef_Cfg_Par
- Amba_AdasFCMD_GetPar
- Amba_AdasMDY_Proc
- Amba_AdasMDY_Init
- Amba_AdasMDY_Deinit
- Amba_AdasMDY_GetDefCfg
- Amba_AdasMDY_SetCfg
- Amba_AdasMDAE_Proc
- Amba_AdasMDAE_Init
- Amba_AdasMDAE_Deinit
- Amba_AdasMDAE_GetDefCfg
- Amba_AdasMDAE_SetCfg

2.1. Amba_Adas_Init

Description:

- This function initializes an instance of ADAS.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**Img**	Image YUV Information
AMBA_ADAS_SCENE_PARAMS_s*	**ScenePar**	ScenePar input scene parameters pointer
AMBA_ADAS_VIEWANGLE_s*	**ViewAngle**	Input ViewAngle parameters pointer

*Table 1: Parameters for ADAS API **Amba_Adas_init**().*

Returns:

Return	Description
0	Success
-1	Initialization failure

*Table 2: Returns for ADAS API **Amba_Adas_Init**().*

2.1.1.Amba_Ada_Init > AMP_ENC_YUV_INFO_s

Field	Description
ChannelID	0: Full-size RGB (interleaved colors)
colorFmt	YUV Color format
uvAddr	UV buffer address
ysize	Y buffer address
pitch	YUV buffer pitch
width	YUV buffer width
height	YUV buffer height

*Table 3: Definition of **Amba_Adas_frame_format_t** for ADAS API **Amba_Adas_init**().*

Examples:

```
int Rval = 0;
    AMBA_ADAS_SCENE_PARAMS_s par;
    AMBA_ADAS_VIEWANGLE_s Vagl;
    AMBA_ADAS_LDW_PARAMS_s ldw_par = {0};
    AMBA_ADAS_FCW_PARAMS_s fcw_par = {0};
    AMBA_ADAS_LDW_OUTPUTEVENT_s LdwEvent = {0};
    AMBA_ADAS_FCW_OUTPUTEVENT_s FcwEvent = {0};
    Amba_Adas_GetSceneParams( &par);
    Amba_Adas_SetSceneParams(&par);
    Rval = Amba_Adas_Init(img, &par, &Vagl);
```

See Also:

None

2.2. Amba_Adas_Proc

Description:

- This function processes the current video frame and gets events if any.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**Img**	Data for the current frames. Please refer to Section 2.1.1 for more details.
AMBA_ADAS_AUX_DATA_s*	**AuxData**	Auxiliary information such as GPS, Dashboard. Dashboard information is not used currently. Please refer to Section 2.2.1 for more details.
AMBA_ADAS_OUTPUTEVENT_s*	**OutEvent**	Information about lane departure event. NULL if no event. Please refer to Section 2.2.2 for more details.
int	**ts**	Time Stamp

*Table 4: Parameters for ADAS API **Amba_Adas_Proc**().*

Returns:

Return	Description
0	Success

*Table 5: Returns for ADAS API **Amba_Adas_Proc**().*

2.2.1.Amba_Adas_Proc > AMBA_ADAS_AUX_DATA_s

Type	Field	Description
Amba_Adas_gps_Info_t*	**pGpsInfo**	GPS information. Please refer to Section 2.2.2 for more details.
Amba_Adas_dashboard_Info_t*	**pDashBoardInfo**	Vehicle dashboard information. Please refer to Section 2.2.3 for more details.

*Table 6: Definition of **Amba_Adas_aux_data_t** for ADAS API **Amba_Adas_Proc**().*

2.2.2.Amba_Adas_Proc > AMBA_ADAS_GPS_INFO_s

Type	Field	Description
float	**Speed**	Vehicle speed in km/hr
float	**Bearing**	Vehicle bearing (Direction of travel) in range 0-360, clockwise with true North being 0

*Table 7: Definition of **Amba_Adas_Gps_Info_t** for ADAS API **Amba_Adas_Pro().***

2.2.3.Amba_Adas_Proc > AMBA_ADAS_DASHBOARD_INFO_s

Type	Field	Description
int	**LeftTurnSignal**	1 if On, 0 if Off, -1 if Unknown
Int	**RightTurnSignal**	1 if On, 0 if Off, -1 if Unknown

Table 8:** Definition of **AMBA_ADAS_DASHBOARD_INFO_s** for ADAS API **Amba_Adas_Pro().

2.2.4.Amba_Adas_Proc > AMBA_ADAS_OUTPUTEVENT_s

Type	Field	Description
AMBA_ADAS_LDW_OUTPUTEVENT_s*	**LdwEvent**	Ldws Event. Please refer to Section 2.2.5 for more details.
AMBA_ADAS_FCW_OUTPUTEVENT_s*	**FcwEvent**	Fcwd Event. Please refer to Section 2.2.11 for more details.
AMBA_ADAS_SCENE_STATUS_s*	**SceneStatus**	Scene status. Please refer to Section 2.5.1 for more details.

*Table 9: Definition of **Amba_Adas_dashboard_Info_t** for ADAS API **Amba_Adas_Pro()**.*

2.2.5.Amba_Adas_Proc > AMBA_ADAS_LDW_OUTPUTEVENT_s

Type	Field	Description
AMBA_ADAS_DEPARTUREDIRECTION	**Direction**	Departure Direction. Please refer to Section 2.2.6 for more details.
AMBA_ADAS_LANEMARKTYPE	**MarkType**	Lane mark Type. Please refer to Section 2.2.7 for more details.

*Table 10: Definition of **AMBA_ADAS_LDW_OUTPUTEVENT_s** for **Amba_Adas_Proc**().*

2.2.6.Amba_Adas_Proc > AMBA_ADAS_DEPARTUREDIRECTION

Field	Description
AMBA_ADAS_DdTowardsLeft	0: Left departure
AMBA_ADAS_DdTowardsRight	1: Right departure

*Table 11: Definition of **AMBA_ADAS_DEPARTUREDIRECTION** for **Amba_Adas_Proc**().*

2.2.7.Amba_Adas_Proc >AMBA_ADAS_LANEMARKTYPE

Type	Field	Description
AMBA_ADAS_LINETYPE	**Type**	Line Type. Please refer to Section 2.2.8 for more details.
AMBA_ADAS_LINESOLIDITY	**Solidity**	Line Solidity. Please refer to Section 2.2.9 for more details.
AMBA_ADAS_LINECOLOR	**Color**	Line Color. Please refer to Section 2.2.10 for more details.

*Table 12: Definition ofAMBA_ADAS_LANEMARKTYPE for **Amba_Adas_Proc**().*

2.2.8.Amba_Adas_Proc > AMBA_ADAS_LINETYPE

Field	Description
AMBA_ADAS_LTUNKNOWN	0: Unknown Type
AMBA_ADAS_LTSINGLE	1: Single line
AMBA_ADAS_LTDOUBLE	2: Double line
AMBA_ADAS_LTROADSIDELEFT	3: Left road side (no lane line)
AMBA_ADAS_LTROADSIDERIGHT	4: Right road side (no lane line)

*Table 13: Definition of **AMBA_ADAS_LINETYPE** for **Amba_Adas_Proc**().*

2.2.9.Amba_Adas_Proc > AMBA_ADAS_LINESOLIDITY

Field	Description
AMBA_ADAS_LSUNKNOWN	0: Unknown Type
AMBA_ADAS_LSSOLID	1: Solid line
AMBA_ADAS_LSDASHED	2: Dashed/broken line

*Table 14: Definition of **AMBA_ADAS_LINESOLIDITY** for ADAS API **Amba_Adas_Proc**().*

2.2.10.Amba_Adas_Proc > AMBA_ADAS_LINECOLOR

Field	Description
AMBA_ADAS_LCUNKNOWN	0: Unknown Color
AMBA_ADAS_LCWHITE	1: White Color
AMBA_ADAS_LCYELLOW	2: Yellow Color

*Table 15: Definition of **AMBA_ADAS_LINECOLOR** for ADAS API **Amba_Adas_Proc**.*

2.2.11.Amba_Adas_Proc > AMBA_ADAS_FCW_OUTPUTEVENT_s

Type	Field	Description
int	**unused**	Currently not in use

*Definition of **Amba_Adas_GetFcwOutput_event_t** for **Amba_Adas_Proc()***

Examples:

```
AMBA_ADAS_GPS_INFO_s gpsInfo;
    AMBA_ADAS_AUX_DATA_s AuxData;
    AMBA_ADAS_LDW_OUTPUTEVENT_s LdwEvent = {0};
    AMBA_ADAS_FCW_OUTPUTEVENT_s FcwEvent = {0};
    AMBA_ADAS_SCENE_STATUS_s SceneStatus = {0};
    AMBA_ADAS_OUTPUTEVENT_s OutEvent = {0};
    unsigned int ts = (unsigned int) AmbaTimer_GetSysTickCount();
    ts = (unsigned int) AmbaTimer_GetSysTickCount();

    if (adas_init == 0) {
        int err = 0;
        err = Init_ADAS( img);
        if (err == 0) {
            adas_init = 1;
        } else {
            AmbaPrintColor(RED, "UT Init_ADAS err = %d \n", err );
            return -1;
        }
    }
    memset(&gpsInfo, 0, sizeof(gpsInfo));
    gpsInfo.Speed = 100;

    memset(&AuxData, 0, sizeof(AuxData));
    AuxData.pGpsInfo = &gpsInfo;

    OutEvent.FcwEvent = &FcwEvent;
    OutEvent.LdwEvent = &LdwEvent;
    OutEvent.SceneStatus = &SceneStatus;

    Amba_Adas_Proc( img, &AuxData, &OutEvent, ts);
    if (OutEvent.LdwEvent != NULL) {
        AmbaPrintColor(RED, "Departure: %s \n",(OutEvent.LdwEvent-
>Direction == AMBA_ADAS_DdTowardsLeft) ? "left" : "right");
    }
    if (OutEvent.FcwEvent != NULL) {
        AmbaPrintColor(RED, "Frontal Collision Warning \n");
    }
```

See Also:

None

2.3. Amba_Adas_SetSceneParams

Description:

- This function sets the scene parameters typically related to calibration.

Parameters:

Type	Parameter	Description
AMBA_ADAS_SCENE_PARAMS_s	**Src**	Scene parameters to be set. Please refer to Section 2.3.1 for more details.

*Table 16: Parameters for ADAS API **Amba_Adas_SetSceneParams**().*

Returns:

Return	Description
0	Success

*Table 17: Returns for ADAS API **Amba_Adas_SetSceneParams**().*

2.3.1.Amba_Adas_SetSceneParams > AMBA_ADAS_SCENE_PARAMS_s

Type	Field	Description
int	**AutomaticCalibration**	Set it to 1 to enable automatic calibration in ADAS. ADAS requires few minutes cumulatively with speed greater than 30 km/hr, to complete automatic calibration
int	**AutoCalibrationActive**	To be used if automaticCalibration=1. Set autoCalibrationActive=1 to enable autocalibration processing; set autoCalibrationActive=0 to disable autocalibration processing. If autoCalibrationActive=0, detected calibration parameters will not be updated over time. Temporarily setting autoCalibrationActive=0 can reduce CPU consumed by ADAS during critical periods of time.
Int	**AutoCalibCpuReduction**	To be used if automaticCalibration=1. Range 1-4. If set to 1 (default), continuous autocalibration will be as fast as possible. If set to 2, continuous autocalibration will be 2 times slower but will take 2 times less CPU. If set to 3,continuous autocalibration will be 3 times slower but will take 3 times less CPU. If set to 4, continuous autocalibration will be 4 times slower but will take 4 times less CPU
float	**HoodLevel**	Level of vehicle hood position, as % of image height, top row being 0. To be used when **AutomaticCalibration = 0**
float	**HorizonLevel**	Level of horizon position, as % of image height, top row being 0. To be used when **AutomaticCalibration = 0**
float	**HorizontalPan**	Horizontal position of point where the lane lines seem to converge, as % of image width, leftmost column being 0. To be used when **AutomaticCalibration = 0**
AMBA_ADAS_CAMERA_MOUNT_HEIGHT_e	**CameraMountHeight**	Mounting height of camera. This should be always set, irrespective of the value of **AutomaticCalibration**. Please refer to Section 2.3.2 for more details.

*Table 18: Definition of **AMBA_ADAS_SCENE_PARAMS_s** for ADAS API **Amba_Adas_SetSceneParams**().*

2.3.2.Amba_Adas_SetSceneParams > AMBA_ADAS_CAMERA_MOUNT_HEIGHT_e

Field	Description
ADAS_CMTGOLFCLASS	0: Sedan Type vehicle (Typical camera height being 1.2 m)
ADAS_CMTCROSSOVER	1: SUV Type vehicle (Typical camera height being 1.35 m)
ADAS_CMTBUSORTRUCK	2: Bus or truck Type vehicle (Typical camera height being 2 m)

*Table 19: Definition of **AMBA_ADAS_CAMERA_MOUNT_HEIGHT_e** for ADAS API **Amba_Adas_SetSceneParams**().*

Examples:

```
AMBA_ADAS_SCENE_PARAMS_s params = {0};
Amba_Adas_GetSceneParams( &params);
Amba_Adas_GetSceneParams( &params);
par.CameraMountHeight = ADAS_CMTGOLFCLASS ;
par.AutoCalibrationActive = 0;
par.AutoCalibCpuReduction = 0;
par.AutomaticCalibration = 0;
par.HoodLevel = VAUT_HoodLevel;
par.HorizonLevel = VAUT_HorizonLevel;
par.HorizontalPan = 50;
Amba_Adas_SetSceneParams(&params);
```

See Also:

None

2.4. Amba_Adas_GetSceneParams

Description:

- This function gets the current scene parameters, including the calibration information.

Parameters:

Type	Parameter	Description
AMBA_ADAS_SCENE_PARAMS_s*	**Des**	Scene parameters to be set. Please refer to Section 2.3.1 for more details.

*Table 20: Parameters for API **Amba_Adas_GetSceneParams**().*

Returns:

Return	Description
0	Success

*Table 21: Returns for ADAS API **Amba_Adas_GetSceneParams**().*

Examples:

```
AMBA_ADAS_SCENE_PARAMS_s params = {0};
int RETURN_VALUE = Amba_Adas_GetSceneParams(p_hInstance, &params);
```

See Also:

None

2.5. Amba_Adas_GetSceneStatus

Description:

- This function gets the current scene status. Primarily used to check if auto-calibration is complete after which ADAS becomes ready to detect FCW and LDW conditions.

Parameters:

Type	Parameter	Description
AMBA_ADAS_SCENE_STATUS_s**	**pp_sceneStatus[out]**	Scene status parameters. Please refer to Section 2.5.1 for more details.

*Table 22: Parameters for API **Amba_Adas_get_scence_status**.*

Returns:

Return	Description
0	Success

*Table 23: Returns for ADAS API **Amba_Adas_GetSceneStatus().***

2.5.1.Amba_Adas_GetSceneStatus > AMBA_ADAS_SCENE_STATUS_s

Type	Field	Description
int	**IsCalibrationDetected**	Whether or not ADAS has completed automatic calibration. If automatic calibration was enabled in AMBA_ADAS_SCENE_PARAMS_s (please refer to Section 2.6.1 for details), then IsCalibrationDetected should be checked to see if calibration is complete. ADAS will not generate any LDW or FCW until IsCalibrationDetected becomes 1.
float	**HoodLevel**	Current value of hood Y position (% of image height, range 0-100, topmost row being 0)
float	**horizontalLevel**	Current value of horizon Y position (% of image height, range 0-100, topmost row being 0)
float	**HorizontalPan**	Current value of pan (i.e. horizon X) (% of image width, range 0-100, leftmost column being 0)
void*	**p_DebugInfo**	Debug information, for internal use only.

*Table 24: Definition of **AMBA_ADAS_SCENE_STATUS_s** for **Amba_Adas_GetSceneStatus**().*

Examples:

```
AMBA_ADAS_SCENE_STATUS_s* p_sceneStatus = NULL;
int RETURN_VALUE = Amba_Adas_GetSceneStatus(p_hInstance, &p_sceneStatus);
if (p_sceneStatus != NULL)
{
    if (p_sceneStatus->IsCalibrationDetected)
    {
        printf("Auto calibration successfull!\n")
    }
}
```

See Also:
None

2.6. Amba_Adas_SetLdwParams

Description:

- This function sets the LDW detection parameters such as sensitivities.

Parameters:

Type	Parameter	Description
AMBA_ADAS_LDW_PARAMS_s	**params[in]**	LDW parameters. Please refer to Section 2.6.1 for more details.

*Table 25: Parameters for API **Amba_Adas_SetLdwParams()**.*

Returns:

Return	Description
0	Success

*Table 26: Returns for ADAS API **Amba_Adas_SetLdwParams**().*

2.6.1.Amba_Adas_SetLdwParams > AMBA_ADAS_LDW_PARAMS_s

Type	Field	Description
int	**IsEnabled**	Set it to 1 to enable LDWS, else set it to 0
AMBA_ADAS_SENSITIVITY_LEVEL_e	**LaneDetectSensitivity**	Lane lines detection sensitivity. Please refer to Section 2.6.2 for more details.
int	**bSeparateLeftRightDepartureSens**	If set to 0, then use only LdwSensitivity. If set to 1, then use LdwLeftSensitivity and LdwRightSensitivity
AMBA_ADAS_SENSITIVITY_LEVEL_e	**LdwSensitivity**	Departure warning sensitivity, common for both left and right departures.
AMBA_ADAS_SENSITIVITY_LEVEL_e	**LdwLeftSensitivity**	Departure warning sensitivity for left departure.
AMBA_ADAS_SENSITIVITY_LEVEL_e	**LdwRightSensitivity**	Departure warning sensitivity for right departure.
float	**MinLdwsSpeedKmph**	Vehicle speed in km/hr above which LDWS should be enabled. Valid range is 10-200 km/hr.

*Table 27: Definition of **AMBA_ADAS_LDW_PARAMS_s** for **Amba_Adas_SetLdwParams**().*

2.6.2.Amba_Adas_SetLdwParams > AMBA_ADAS_SENSITIVITY_LEVEL_e

Field	Description
ADAS_SL_LOW	0: Low sensitivity
ADAS_SL_MEDIUM	1: Medium sensitivity
ADAS_SL_HIGH	2: High sensitivity

*Table 28: Definition of **AMBA_ADAS_SENSITIVITY_LEVEL_e** for **Amba_Adas_SetLdwParams**().*

Examples:

```
AMBA_ADAS_LDW_PARAMS_s params = {0};
int RETURN_VALUE = Amba_Adas_GetLdwParams(p_hInstance, &params);
params.IsEnabled = 1; // modify
params.MinLdwsSpeedKmph = 30.0f;
// leave other parameters default
RETURN_VALUE = Amba_Adas_SetLdwParams(p_hInstance, params);
```

See Also:

None

2.7. Amba_Adas_GetLdwParams

Description:

- This function gets the current LDW parameters.

Parameters:

Type	Parameter	Description
AMBA_ADAS_LDW_PARAMS_s*	**p_params[out]**	LDW parameters. Please refer to Section 2.6.1 for more details.

*Table 29: Parameters for API **Amba_Adas_GetLdwParams()**.*

Returns:

Return	Description
0	Success

*Table 30: Returns for ADAS API **Amba_Adas_GetLdwParams**().*

Examples:

```
AMBA_ADAS_LDW_PARAMS_s params = {0};
int RETURN_VALUE = Amba_Adas_GetLdwParams( &params);
```

See Also:

None

2.8. Amba_Adas_GetLdwOutput

Description:

- This function gets the additional information from lane departure system such as the information about the lane.

Parameters:

Type	Parameter	Description
AMVA_ADAS_LDW_OUTPUT_s**	**pp_ldwOutput[out]**	Additional information from the lane departure system. Please refer to Section 2.8.1 for more details.

*Table 31: Parameters for API **Amba_Adas_GetLdwOutput()**.*

Returns:

Return	Description
0	Success

*Table 32: Returns for ADAS API **Amba_Adas_GetLdwOutput**().*

2.8.1.Amba_Adas_GetLdwOutput > AMVA_ADAS_LDW_OUTPUT_s

Type	Field	Description
AMBA_ADAS_RoadMarkLine_s	**LeftLine**	Information about left line. Please refer to Section 2.8.2 for more details.
AMBA_ADAS_RoadMarkLine_s	**RightLine**	Information about right line

*Table 33: Definition of **AMBA_ADAS_LDW_OUTPUT_s** for **Amba_Adas_GetLdwOutput**().*

2.8.2.Amba_Adas_GetLdwOutput > AMBA_ADAS_RoadMarkLine_s

Type	Field	Description
int	**IsDetected**	Whether the lane line was detected
AMBA_ADAS_LANEMARKTYPE	**MarkType**	Information about lane mark. Please refer to Section 2.2.7 for more details.
AMBA_VA_POINT_s*	**p_Points**	Array of points forming the line. Please refer to Section 2.8.3 for more details.
int	**PointsCount**	Number of points in **p_Points**.

*Table 34: Definition of **AMBA_ADAS_RoadMarkLine_s** for **Amba_Adas_GetLdwOutput**().*

2.8.3.Amba_Adas_GetLdwOutput > AMBA_VA_POINT_s

Type	Field	Description
float	**X**	X-coordinate
float	**Y**	Y-coordinate

*Table 35: Definition of **AMBA_VA_POINT_s** for **Amba_Adas_GetLdwOutput**().*

Examples:

```
AMVA_ADAS_LDW_OUTPUT_s* p_ldwOutput = NULL;
int RETURN_VALUE = Amba_Adas_GetLdwOutput(p_hInstance, &p_ldwOutput);
if (p_ldwOutput != NULL)
{
     Draw(p_ldwOutput->LeftLine);
     Draw(p_ldwOutput->RightLine);
}
```

2.9. Amba_Adas_SetFcwParams

Description:

- This function sets the FCW detection parameters such as sensitivities.

Parameters:

Type	Parameter	Description
AMBA_ADAS_FCW_PARAMS_s	**params[in]**	FCW parameters. Please refer to Section 2.9.1 for more details.

*Table 36: Parameters for API **Amba_Adas_SetFcwParams**().*

Returns:

Return	Description
0	Success

*Table 37: Returns for ADAS API **Amba_Adas_SetFcwParams**().*

2.9.1.Amba_Adas_SetFcwParams > AMBA_ADAS_FCW_PARAMS_s

Type	Field	Description
int	**IsEnabled**	Set it to 1 to enable FCWS, else set it to 0
AMBA_ADAS_SENSITIVITY_LEVEL_e	**VehicleDetectSensitivity**	Sensitivity for vehicle detection (*Not used currently*). Please refer to Section 2.6.2 for more details.
AMBA_ADAS_SENSITIVITY_LEVEL_e	**FcwSensitivity**	Sensitivity level for FCW. Please refer to Section 2.6.2 for more details.

*Table 38: Definition of **AMBA_ADAS_FCW_PARAMS_s** for **Amba_Adas_SetFcwParams**().*

Examples:

```
AMBA_ADAS_FCW_PARAMS_s params = {0};
int RETURN_VALUE = Amba_Adas_GetFcwParams(&params);
params.IsEnabled = 1; // modify
// leave other parameters default
RETURN_VALUE = Amba_Adas_SetFcwParams(params);
```

See Also:

None

2.10. Amba_Adas_GetFcwParams

Description:

- This function gets the current FCW parameters.

Parameters:

Type	Parameter	Description
AMBA_ADAS_FCW_PARAMS_s*	**p_params[out]**	FCW parameters. Please refer to Section 2.9.1 for details.

*Table 39: Parameters for API **Amba_Adas_GetFcwParams()**.*

Returns:

Return	Description
0	Success

*Table 40: Returns for ADAS API **Amba_Adas_GetFcwParams**().*

Examples:

```
AMBA_ADAS_FCW_PARAMS_s params = {0};
int RETURN_VALUE = Amba_Adas_GetFcwParams(&params);
```

See Also:

None

2.11. Amba_Adas_GetFcwOutput

Description:

- This function gets the additional information from the frontal collision system, such as the information about the front vehicle.

Parameters:

Type	Parameter	Description
AMBA_ADAS_FCW_OUTPUT_s**	**pp_fcwOutput[out]**	Additional Information from frontal collision system. Please refer to Section 2.11.1 for more details.

*Table 41: Parameters for API **Amba_Adas_GetFcwOutput()**.*

Returns:

Return	Description
0	Success

*Table 42: Returns for ADAS API **Amba_Adas_GetFcwParams**().*

2.11.1.Amba_Adas_GetFcwOutput > AMBA_ADAS_FCW_OUTPUT_s

Type	Field	Description
int	**FcwActive**	Tells if FCW is active. FCWS can suppress FCW in certain conditions.
AMBA_VA_POINT_s*	**p_FrontVehiclePoints**	Array of points representing the front vehicle in the image. NULL if no vehicle detected. Please refer to Section 2.11.3 for more details.
int	**FrontVehiclePointsCount**	Number of points in **p_FrontVehiclePoints**
float	**FrontVehicleDistance**	Front vehicle distance in meters. -1 if no vehicle is detected.
float	**FrontVehicleTtc**	The minimum of TTC and Headway, in secs. -1 if no vehicle is detected.
AMBA_ADAA_FCW_WARNING_LEVEL_e	**WarningLevel**	Warning level based on **FrontVehicleTtc**. Please refer to Section 2.14.2 for more details.
void*	**p_DebugInfo**	Debug information, for internal use only.

*Table 43: Definition of **AMBA_ADAS_FCW_OUTPUT_s** for **Amba_Adas_GetFcwOutput**().*

2.11.2.Amba_Adas_GetFcwOutput > AMBA_ADAA_FCW_WARNING_LEVEL_e

Field	Description
ADAS_FWL_NONE	0: No warning, for e.g. if there is no front vehicle or the front vehicle is far
ADAS_FWL_LOW	1: Low level of FCW
ADAS_FWL_MEDIUM	2: Medium level of FCW
ADAS_FWL_HIGH	3: High level of FCW

*Table 44: Definition of **AMBA_ADAA_FCW_WARNING_LEVEL_e** for **Amba_Adas_GetFcwOutput**().*

Examples:

```
AMBA_ADAS_FCW_OUTPUT_s* p_fcwOutput = NULL;
int RETURN_VALUE = Amba_Adas_GetFcwOutput(&p_fcwOutput);
if (p_fcwOutput != NULL)
{
     printf("TTC=%d\n", p_fcwOutput->FrontVehicleTtc);
}
```

See Also:
None

2.12. Amba_AdasLLWS_Proc

Description:

- This function does the low light warning process.

Parameters:

Type	Parameter	Description
AMBA_DSP_EVENT_CFA_3A_DATA_s*	**pData3A**	3A information data from 3A data handler
AMP_LLWS_PAR_t*	**Ppar**	Additional Information from the frontal collision system. Please refer to Section 2.12.1 for more details.
int*	**pLLWSEvent**	1 is low light detection. 0 for Success.

*Table 45: Parameters for API 2.12. **Amba_AdasLLWS_Proc()**.*

Returns:

Return	Description
0	OK
-1	Failure due to parameter errors

*Table 46: Returns for ADAS API **Amba_AdasLLWS_Proc()**.*

Examples:

```
int rval = 0;
AMBA_DSP_EVENT_CFA_3A_DATA_s* pdata = input;
int llws_event = 0;
AMP_LLWS_CFG_t cfg = {0};
AMP_LLWS_PAR_t par = {0};
/// set config
cfg.LLWSSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
/// set pars
par.HoodLevel = DEFAULT_HOODLEVEL;
par.HorizonLevel = DEFAULT_HORIZLEVEL;
if (llws_init == 0) {
      rval = Amba_AdasLLWS_SetCfg(&cfg);
      rval |= Amba_AdasLLWS_Init(&par);
      if (rval == 0) {
            llws_init = 1;
}
 AmbaPrintColor(rval, "Amba_AdasLLWS_Init  return %d", rval);
}
if (llws_init == 1) {
      rval = Amba_AdasLLWS_Proc(pdata, &par, &llws_event);
      if (rval < 0) {
      llws_init = 0;
      }
}

if (llws_event == AMBA_LLWS_LOW_LIGHT) {
  AmbaPrintColor(RED, "Turn  on the light”);
  AmbaPrintColor(RED, "Turn  on the light”);

}
```

See Also:

None

2.12.1.Amba_AdasLLWS_Proc > AMP_LLWS_PAR_t

Type	Field	Description
float	**HoodLevel**	Set it to 1 to enable LLWS, else set it to 0
float	**HorizonLevel**	Level of vehicle hood position, as % of image height, top row being 0.
int	**IsCalibrationDetected**	Whether or not ADAS has completed automatic calibration. If automatic calibration was enabled in AMBA_ADAS_SCENE_PARAMS_s (please refer to section 2.6.1 for details), then IsCalibrationDetected should be checked to see if calibration is complete. ADAS will not generate any LDW or FCW until IsCalibrationDetected becomes 1.
int	**IsUpdate**	Update the parameters.

*Table 47: Definition of **AMP_LLWS_PAR_t** for **Amba_AdasLLWS_Proc**().*

2.13. Amba_AdasLLWS_Init

Description:

- This function initializes LLWS buffer and sets default parameters.

Parameters:

Type	Parameter	Description
AMP_LLWS_PAR_t	**pPar**	LLW parameters. Please refer to Section 2.12.1 for more details.

*Table 48: Parameters for ADAS API **Amba_Adas_set_llw_params**().*

Returns:

Return	Description
0	Success

*Table 49: Returns for ADAS API **Amba_AdasLLWS_Init**().*

Examples:

```
AMP_LLWS_CFG_t cfg = {0};
AMP_LLWS_PAR_t par = {0};
/// set config
cfg.LLWSSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
/// set pars
par.HoodLevel = DEFAULT_HOODLEVEL;
par.HorizonLevel = DEFAULT_HORIZLEVEL;
if (llws_init == 0) {
      rval = Amba_AdasLLWS_SetCfg(&cfg);
      rval |= Amba_AdasLLWS_Init(&par);
}
```

See Also:

None

2.14. Amba_AdasLLWS_Deinit

Description:

- This function resets the LLWS system configuration and parameters .

Returns:

Return	Description
0	Success

*Table 50: Returns for ADAS API **Amba_AdasLLWS_Deinit**().*

Examples:

```
Amba_AdasLLWS_Deinit();
```

See Also:

None

2.15. Amba_AdasLLWS_SetCfg

Description:

- This function gets the low light warning parameters.

Parameters:

Type	Parameter	Description
AMP_LLWS_CFG_t*	**pCfg**	LLW parameters. Please refer to Section 2.15.1 for more details.

*Table 51: Parameters for ADAS API **Amba_AdasLLWS_SetCfg**().*

Returns:

Return	Description
0	Success

Table 52: Returns for ADAS API Amba_AdasLLWS_SetCfg().

Examples:

```
AMP_LLWS_PAR_t params = {0};
int RETURN_VALUE = Amba_Adas_get_llw_params(&params);
```

2.15.1.Amba_AdasLLWS_SetCfg > AMP_LLWS_CFG_t

Type	Field	Description
int	**IsEnabled**	Enable LLWS processing.
AMBA_ADAS_SENSITIVITY_LEVEL_e	**LLWSSensitivity**	LLW parameters. Please refer to Section 2.6.2 for more details.

*Table 53: Definition of **AMP_LLWS_PAR_t** for **Amba_AdasLLWS_Proc**().*

See Also:
None

2.16. Amba_AdasLLWS_GetDefCfg

Description:

- This function sets scene parameters typically related to calibration.

Parameters:

Type	Parameter	Description
AMP_LLWS_CFG_t*	**pCfg**	Additional information from LLWS. Please refer to Section 2.15.1 for more details.

*Table 54: Parameters for ADAS API **Amba_AdasLLWS_GetDefCfg**().*

Returns:

Return	Description
0	Success

*Table 55: Returns for ADAS API **Amba_AdasLLWS_GetDefCfg**().*

Examples:

```
AMP_LLWS_CFG_t llwOutput[1];
int RETURN_VALUE = Amba_AdasLLWS_GetDefCfg(&llwOutput);
```

See Also:

None

2.17. Amba_AdasLLWS_GetPar

Description:

- This function gets the low light warning parameters.

Parameters:

Type	Parameter	Description
AMP_LLWS_PAR_t*	**pPar**	LLW parameters. Please refer to Section 2.12.1 for more details.

*Table 56: Parameters for ADAS API **Amba_AdasLLWS_GetPar**().*

Returns:

Return	Description
0	Success

*Table 57: Returns for ADAS API **Amba_AdasLLWS_GetPar**().*

Examples:

```
AMP_LLWS_PAR_t params = {0};
int RETURN_VALUE = Amba_AdasLLWS_GetPar(&params);
```

See Also:

None

2.18. Amba_AdasFCMD_Proc

Description:

- This function does low light warning process.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**img**	Additional Information from frontal collision system. Please refer to Section 2.1.1 for more details.
AMP_FCMD_PAR_t*	**pPar**	Additional Information from frontal collision system. Please refer to Section 2.18.1 for more details.
AMBA_ADAS_FCMD_EVENT_e*	**pFCMDEvent**	1 is for low light detection. 0 is for Success.

*Table 58: Parameters for API **Amba_AdasFCMD_Proc()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors

*Table 59: Returns for ADAS API **Amba_AdasFCMD_Proc()**.*

Examples:

```
AMP_FCMD_CFG_t cfg = {0};
AMP_FCMD_PAR_t par = {0};
if (fcmd_init == 1) {
      mwut_gps.Speed = (float)gps_sim[gps_index];
      par.IsGPSandDashboard = 1;
      par.Aux_data.pGpsInfo = &mwut_gps;
      gps_index ++;
      if (gps_index == 35 ) {
            gps_index = 0;
}
      rval = Amba_AdasFCMD_Proc(img, &par, &fcmd_event);
      //AmbaPrintColor(rval, "Amba_AdasFCMD_Proc return %d", rval);
      if (rval < 0) {
                  fcmd_init = 0;
      }
}
cfg.FCMDSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
 /// set pars
par.HoodLevel = DEFAULT_HOODLEVEL;
par.HorizonLevel = DEFAULT_HORIZLEVEL;
par.IsUpdate = 1;

if (fcmd_event == AMBA_FCMD_MOVE) {
      AmbaPrintColor(RED, "Frontal Vehicle Move \n");
      AmbaKAL_TaskSleep(3000);
}
```

See Also:
None

2.18.1.Amba_AdasFCMD_Proc > AMP_FCMD_PAR_t

Type	Field	Description
float	**HoodLevel**	Set it to 1 to enable LLWS, else set it to 0.
float	**HorizonLevel**	Level of vehicle hood position, as % of image height, top row being 0.
int	**IsCalibrationDetected**	Checks whether or not ADAS has completed automatic calibration. If automatic calibration was enabled in AMBA_ADAS_SCENE_PARAMS_s (please refer to Section 2.6.1 for details), then IsCalibrationDetected should be checked to see if calibration is complete. ADAS will not generate any LDW or FCW until IsCalibrationDetected becomes 1.
int	**BoundingSky**	The ROI TOP bounding
int	**BoundingHood**	The ROI BOT bounding
int	**BoundingLeft**	The ROI left bounding
int	**BoundingRight**	The ROI right bounding
int	**BoundingH**	The ROI height
int	**BoundingW**	The ROI width
int	**IsUpdate**	Update the parameters.
AMBA_ADAS_AUX_DATA_s	**Aux_data**	Vehicle dashboard information. Please refer to Section 2.2.1 for more details.
int	**IsGPSandDashboard**	Update the Aux_Data

*Table 60: Definition of **AMP_FCMD_PAR_t** for **Amba_AdasFCMD_Proc**().*

See Also:
None

2.19. Amba_AdasFCMD_Init

Description:

- This function initializes LLWS buffer and sets default parameters.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**img**	
AMP_FCMD_PAR_t*	**pPar**	FCMD parameters. Please refer to Section 2.18.1 for more details.

*Table 61: Parameters for ADAS API **Amba_Adas_set_llw_params**().*

Returns:

Return	Description
0	Success
-4	Amba_AdasFCMD_Init memory is not allocated

*Table 62: Returns for ADAS API **Amba_AdasFCMD_Init**().*

Examples:

```
AMP_FCMD_CFG_t cfg = {0};
AMP_FCMD_PAR_t par = {0};
cfg.FCMDSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
 /// set pars
 par.HoodLevel = DEFAULT_HOODLEVEL;
 par.HorizonLevel = DEFAULT_HORIZLEVEL;
 par.IsUpdate = 1;

 rval = Amba_AdasFCMD_SetCfg(&cfg);
 if ( rval != OK){
      AmbaPrint("Amba_AdasFCMD_SetCfg failed");
      return 0;
 }
 rval = Amba_AdasFCMD_Init(img, &par);
```

See Also:

None

2.20. Amba_AdasFCMD_Deinit

Description:

- This function resets the FCMD system configuration and parameters .

Returns:

Return	Description
0	Success

*Table 63: Returns for ADAS API **Amba_AdasFCMD_Deinit**().*

Examples:

```
Amba_AdasFCMD_Deinit();
```

See Also:

None

2.21. Amba_AdasFCMD_SetCfg

Description:

- This function gets the low light warning parameters.

Parameters:

Type	Parameter	Description
AMP_FCMD_CFG_t*	**pCfg**	FCMD parameters. Please refer to Section 2.15.1 for more details.

*Table 64: Parameters for ADAS API **Amba_AdasFCMD_SetCfg**().*

Returns:

Return	Description
0	Success

*Table 65: Returns for ADAS API **Amba_AdasFCMD_SetCfg**().*

Examples:

```
AMP_FCMD_CFG_t cfg = {0};
AMP_FCMD_PAR_t par = {0};
cfg.FCMDSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
 /// set pars
 par.HoodLevel = DEFAULT_HOODLEVEL;
 par.HorizonLevel = DEFAULT_HORIZLEVEL;
 par.IsUpdate = 1;

 rval = Amba_AdasFCMD_SetCfg(&cfg);
```

See Also:

None

2.21.1.Amba_AdasLLWS_SetCfg > AMP_FCMD_CFG_t

Type	Field	Description
int	**IsEnabled**	Enable FCMD processing.
AMBA_ADAS_SENSITIVITY_LEVEL_e	**FCMDSensitivity**	FCMD sensitivity parameters. Please refer to Section 2.6.2 for more details.
UINT32	**Buf_Size**	Buffer size
AMBA_MEM_CTRL_s	**Buf**	System buffer pointer

*Table 66: Definition of **AMP_LLWS_PAR_t** for **Amba_AdasFCMD_SetCfg**().*

2.22. Amba_AdasFCMD_GetDef_Cfg_Par

Description:

- This function sets scene parameters typically related to calibration.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**img**	Image data. Please refer to Section 2.1.1 for more details.
AMP_FCMD_CFG_t*	**pCfg**	FCMD configuration. Please refer to Section 2.15.1 for more details.
AMP_FCMD_PAR_t*	**pPar**	FCMD parameters. Please refer to Section 2.18.1 for more details.

*Table 67: Parameters for ADAS API **Amba_AdasFCMD_GetDef_Cfg_Par**().*

Returns:

Return	Description
0	Success

Table 68: Returns for ADAS API Amba_AdasFCMD_GetDef_Cfg_Par().

Examples:

```
AMP_LLWS_CFG_t llwOutput[1];
int RETURN_VALUE = Amba_AdasLLWS_GetDefCfg(&llwOutput);
```

See Also:

None

2.23. Amba_AdasFCMD_GetPar

Description:

- This function gets the low light warning parameters.

Parameters:

Type	Parameter	Description
AMP_FCMD_PAR_t*	**pPar**	FCMD parameters. Please refer to Section 2.18.1 for more details.

*Table 69: Parameters for ADAS API **Amba_AdasFCMD_GetPar**().*

Returns:

Return	Description
0	Success

*Table 70: Returns for ADAS API **Amba_AdasFCMD_GetPar**().*

Examples:

```
AMP_FCMD_PAR_t params = {0};
int RETURN_VALUE = Amba_AdasFCMD_GetPar(&params);
```

See Also:

None

2.24. Amba_AdasMDY_Proc

Description:

- This function does motion detection process based on Y data.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**img**	Image YUV Information. Please refer to Section 2.1.1 for more details.
AMBA_MDY_PAR_t*	**pPar**	Motion detection process motion parameters. Please refer to Section 2.24.1 for more details.
int*	**pMDEvent**	1 is motion detection detected. 0 is for Success.

*Table 71: Parameters for API **Amba_AdasMDY_Proc()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors

*Table 72: Returns for ADAS API **Amba_AdasMDY_Proc()**.*

Examples:

```
AMBA_MDY_CFG_t cfg = {0};
AMBA_MDY_PAR_t par = {0};
AMP_ENC_YUV_INFO_s Img = {0};
/// set pars
par.Method = MDY_DEFAULT;
if (mdy_init == 0) {
    Amba_AdasMDY_GetDefCfg( &Img, &cfg);
    rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDYBuf), cfg.MDYBuf_Size, 32);
    if (rval != OK) {
        AmbaPrint(" Process_MDY Can't allocate memory.");
        return 0;
    }
    /// set config
    cfg.RoiData[0].Location.X = 0;
    cfg.RoiData[0].Location.Y = 0;
    cfg.RoiData[0].Location.W = 300;
    cfg.RoiData[0].Location.H = 200;
    cfg.RoiData[0].Sensitivity = 10;
    rval = Amba_AdasMDY_SetCfg(&cfg);
    par.IsUpdate = 1;
    rval |= Amba_AdasMDY_Init(&par);
    if (rval == 0) {
        mdy_init = 1;

}
if (mdy_init == 1) {
    rval = Amba_AdasMDY_Proc(img, &par, &md_event);
    if (rval < 0) {
        mdy_init = 0;
    }
}
```

```
if (md_event == AMBA_MDY_MOVE_DET) {
    AmbaPrintColor(RED, "AMBA_MDY_MOVE_DET***** \n");
}
```

See Also:
None

2.24.1.Amba_AdasMDY_Proc > AMBA_MDY_PAR_t

Type	Field	Description
AMBA_MDY_METHOD_e	**Method**	Method selection. Please refer to Section 2.24.2 for more details.
AMBA_MD_ROI_STATUS_t	**Roi_Status[MOTION_DETECT_ROI_MAX];**	Region of interest setting. Please refer to Section 2.24.3 for more details.
int	**IsUpdate**	Update the parameters.

*Table 73: Definition of **AMP_MDY_PAR_t** for **Amba_AdasMDY_Proc**().*

2.24.2.Amba_AdasMDY_Proc > AMBA_MDY_METHOD_e

Type	Field	Description
int	**MDY_DEFAULT**	DEFAULT algorithm
int	**MDY_MSE**	MSE algorithm

*Table 74: Definition of **AMP_MDY_METHOD_e** for **Amba_AdasMDY_Proc**().*

2.24.3.Amba_AdasMDY_Proc > AMBA_MD_ROI_STATUS_t

Type	Field	Description
AMBA_MOTION_FLAGS_e	**Motion_Flags**	Motion indicators. Please refer to Section 2.24.4 for more details.
AMBA_MOTION_EVENT_e	**Motion_Event**	Motion events. Please refer to Section 2.24.5 for more details.

*Table 75: Definition of **AMBA_MD_ROI_STATUS_t** for **Amba_AdasMDY_Proc**().*

2.24.4.Amba_AdasMDY_Proc > AMBA_MOTION_FLAGS_e

Type	Field	Description
int	**MD_NO_MOTION**	No Motion
int	**MD_IN_MOTION**	In Motion

*Table 76: Definition of **AMBA_MOTION_FLAGS_e** for **Amba_AdasMDY_Proc**().*

2.24.5.Amba_AdasMDY_Proc > AMBA_MOTION_EVENT_e

Type	Field	Description
int	**MD_MOTION_NO_EVENT**	No event
int	**MD_MOTION_START**	Motion start
int	**MD_MOTION_END**	Motion end

*Table 77: Definition of **AMBA_MOTION_EVENT_e** for **Amba_AdasMDY_Proc**().*

2.25. Amba_AdasMDY_Init

Description:

- This function initializes the motion detection process and accepts the allocated buffer.

Parameters:

Type	Parameter	Description
AMBA_MDY_PAR_t*	**pPar**	Additional Information motion detection initialization. Please refer to Section 2.24.1 for more details.

*Table 78: Parameters for API **Amba_AdasMDY_Init()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors
-5	MDY Buffer is NULL

*Table 79: Returns for ADAS API **Amba_AdasMDY_Init()**.*

Examples:

```
AMBA_MDY_CFG_t cfg = {0};
AMBA_MDY_PAR_t par = {0};
AMP_ENC_YUV_INFO_s Img = {0};
/// set pars
par.Method = MDY_DEFAULT;
if (mdy_init == 0) {
    Amba_AdasMDY_GetDefCfg( &Img, &cfg);
    rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDYBuf), cfg.MDYBuf_Size, 32);
    if (rval != OK) {
        AmbaPrint(" Process_MDY Can't allocate memory.");
        return 0;
    }
    /// set config
    cfg.RoiData[0].Location.X = 0;
    cfg.RoiData[0].Location.Y = 0;
    cfg.RoiData[0].Location.W = 300;
    cfg.RoiData[0].Location.H = 200;
    cfg.RoiData[0].Sensitivity = 10;
    rval = Amba_AdasMDY_SetCfg(&cfg);
    par.IsUpdate = 1;
    rval |= Amba_AdasMDY_Init(&par);
    if (rval == 0) {
        mdy_init = 1;

}
```

See Also:

None

2.26. Amba_AdasMDY_Deinit

Description:

- This function deinitializes motion detection and free buffer .

Parameters:

void

Returns:

void

Examples:

```
AMBA_MDY_CFG_t cfg = {0};
AMBA_MDY_PAR_t par = {0};
AMP_ENC_YUV_INFO_s Img = {0};
/// set pars
par.Method = MDY_DEFAULT;
if (mdy_init == 0) {
    Amba_AdasMDY_GetDefCfg( &Img, &cfg);
    rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDYBuf), cfg.MDYBuf_Size, 32);
    if (rval != OK) {
        AmbaPrint(" Process_MDY Can't allocate memory.");
        return 0;
    }
    /// set config
    cfg.RoiData[0].Location.X = 0;
    cfg.RoiData[0].Location.Y = 0;
    cfg.RoiData[0].Location.W = 300;
    cfg.RoiData[0].Location.H = 200;
    cfg.RoiData[0].Sensitivity = 10;
    rval = Amba_AdasMDY_SetCfg(&cfg);
    par.IsUpdate = 1;
    rval |= Amba_AdasMDY_Init(&par);
    if (rval == 0) {
        mdy_init = 1;
  }
  Amba_AdasMDY_Deinit();
```

See Also:

None

2.27. Amba_AdasMDY_GetDefCfg

Description:

- This function performs motion detection process based on Y data.

Parameters:

Type	Parameter	Description
AMP_ENC_YUV_INFO_s*	**img**	Image YUV Information. Please refer to Section 2.1.1 for more details.
AMBA_MDY_CFG_t*	**pCfg**	Motion detection Configuration. Please refer to Section 2.24.1 for more details.

*Table 80: Parameters for API **Amba_AdasMDY_GetDefCfg()**.*

Returns:

void

Examples:

```
AMBA_MDY_CFG_t cfg = {0};
AMBA_MDY_PAR_t par = {0};
AMP_ENC_YUV_INFO_s Img = {0};
/// set pars
par.Method = MDY_DEFAULT;
if (mdy_init == 0) {
    Amba_AdasMDY_GetDefCfg( &Img, &cfg);
    rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDYBuf), cfg.MDYBuf_Size, 32);
    if (rval != OK) {
        AmbaPrint(" Process_MDY Can't allocate memory.");
        return 0;
    }
    /// set config
    cfg.RoiData[0].Location.X = 0;
    cfg.RoiData[0].Location.Y = 0;
    cfg.RoiData[0].Location.W = 300;
    cfg.RoiData[0].Location.H = 200;
    cfg.RoiData[0].Sensitivity = 10;
    rval = Amba_AdasMDY_SetCfg(&cfg);
    par.IsUpdate = 1;
    rval |= Amba_AdasMDY_Init(&par);
    if (rval == 0) {
        mdy_init = 1;

}
```

See Also:

None

2.27.1.Amba_AdasMDY_GetDefCfg > AMBA_MDY_CFG_t

Type	Field	Description
AMBA_MDY_ROI_DATA_t	**RoiData[MOTION_DETECT _ROI_MAX]**	The Region of interest (ROI) Setting. Please refer to Section 2.27.2 for more details.
UINT32	**MDYBuf_Size**	Buffer size
AMBA_MEM_CTRL_s	**MDYBuf**	System buffer pointer

*Table 81: Definition of **AMBA_MDY_CFG_t** for **Amba_AdasMDY_Proc**().*

2.27.2.Amba_AdasMDY_GetDefCfg > AMBA_MDY_ROI_DATA_t

Type	Field	Description
UINT8	**Valid**	On (1)or OFF(0) this ROI.
UINT8	**Sensitivity**	Sensitivity
UINT16	**Luma_Diff_Threshold**	Luma Difference Threshold
AMBA_VA_ROI_s	**Location**	The region of interest (ROI) description

*Table 82: Definition of **AMBA_MDY_ROI_DATA_t** for **Amba_AdasMDY_Proc**().*

2.28. Amba_AdasMDY_SetCfg

Description:

- This function sets the configuration of the motion detection process.

Parameters:

Type	Parameter	Description
AMBA_MDY_CFG_t*	**pCfg**	Motion detection Configuration. Please refer to Section 2.24.1 for more details.

*Table 83: Parameters for API **Amba_AdasMDY_SetCfg()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors

*Table 84: Returns for ADAS API **Amba_AdasMDY_SetCfg()**.*

Examples:

```
AMBA_MDY_CFG_t cfg = {0};
AMBA_MDY_PAR_t par = {0};
AMP_ENC_YUV_INFO_s Img = {0};
/// set pars
par.Method = MDY_DEFAULT;
if (mdy_init == 0) {
    Amba_AdasMDY_GetDefCfg( &Img, &cfg);
    rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDYBuf), cfg.MDYBuf_Size, 32);
    if (rval != OK) {
        AmbaPrint(" Process_MDY Can't allocate memory.");
        return 0;
    }
    /// set config
    cfg.RoiData[0].Location.X = 0;
    cfg.RoiData[0].Location.Y = 0;
    cfg.RoiData[0].Location.W = 300;
    cfg.RoiData[0].Location.H = 200;
    cfg.RoiData[0].Sensitivity = 10;
    rval = Amba_AdasMDY_SetCfg(&cfg);
    par.IsUpdate = 1;
    rval |= Amba_AdasMDY_Init(&par);
    if (rval == 0) {
        mdy_init = 1;

}
```

See Also:

None

2.29. Amba_AdasMDAE_Proc

Description:

- This function performs motion detection process based on Y data.

Parameters:

Type	Parameter	Description
AMBA_DSP_EVENT_CFA_3A_DATA_s*	**pData3A**	3A information data from 3A data handler.
AMBA_MDAE_PAR_t*	**pPar**	Motion detection process motion parameters. Please refer to Section 2.29.1 for more details.
int*	**pMDEvent**	0 is Success. 1 is MD detected.

*Table 85: Parameters for API **Amba_AdasMDAE_Proc()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors

*Table 86: Returns for ADAS API **Amba_AdasMDAE_Proc()**.*

Examples:

```
    AMBA_MDAE_CFG_t cfg = {0};
    AMBA_MDAE_PAR_t par = {0};

    /// set pars
    par.Method = MDAE_DEFAULT;

    if (mdae_init == 0) {
        Amba_AdasMDAE_GetDefCfg(&cfg);
        rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDAEBuf), cfg.MDAEBuf_Size,
32);
        if (rval != OK) {
            AmbaPrint(" Process_MDAE Can't allocate memory.");
            return 0;
        }
        cfg.RoiData[0].Location.X = 0;
        cfg.RoiData[0].Location.Y = 0;
        cfg.RoiData[0].Location.W = 12;
        cfg.RoiData[0].Location.H = 8;
        cfg.RoiData[0].Sensitivity = 2; /// Sensitivity is Multiplier
        cfg.RoiData[0].Threshold = 3900;
        cfg.RoiData[0].Update_Freq = 1;
        cfg.RoiData[0].Update_Cnt = cfg.RoiData[0].Update_Freq - 1;
        rval = Amba_AdasMDAE_SetCfg(&cfg);
        AmbaPrint("Amba_AdasMDAE_SetCfg  %d\n", rval);
        par.IsUpdate = 1;
        rval = Amba_AdasMDAE_Init(&par);
        AmbaPrint("Amba_AdasMDAE_Init  %d\n", rval);
        if (rval == 0) {
            mdae_init = 1;
        }
        AmbaPrintColor(rval, "Amba_AdasMDAE_Init  return %d", rval);
    }
```

```
if (mdae_init == 1) {
    rval = Amba_AdasMDAE_Proc(pdata, &par, &md_event);
    //AmbaPrintColor(rval, "Amba_AdasFCMD_Proc  return %d", rval);
    if (rval < 0) {
        mdae_init = 0;
    }
}

if (md_event == AMBA_MDAE_MOVE_DET) {
    AmbaPrintColor(RED, "AMBA_MDY_MOVE_DET********** \n");
}
```

See Also:
None

2.29.1.Amba_AdasMDAE_Proc > AMBA_MDAE_PAR_t

Type	Field	Description
AMBA_MDAE_METHOD_e	**Method**	Method selection. Please refer to Section 2.29.2 for more details.
AMBA_MD_ROI_STATUS_t	**Roi_Status[MOTION_DETECT_ROI_MAX];**	Region of interest setting. Please refer to Section 2.24.3 for more details.
int	**IsUpdate**	Update the parameters.

*Table 87: Definition of **AMP_MDAE_PAR_t** for **Amba_AdasMDAE_Proc**().*

2.29.2.Amba_AdasMDAE_Proc > AMBA_MDAE_METHOD_e

Type	Field	Description
int	**MDAE_DEFAULT**	DEFAULT algorithm

*Table 88: Definition of **AMP_MDAE_METHOD_e** for **Amba_AdasMDAE_Proc**().*

2.30. Amba_AdasMDAE_Init

Description:

- This function initializes the motion detection process and accepts the allocated buffer.

Parameters:

Type	Parameter	Description
AMBA_MDAE_PAR_t*	**pPar**	Additional Information on motion detection initialization. Please refer to Section 2.29.1 for more details.

*Table 89: Parameters for API **Amba_AdasMDAE_Init()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors
-5	MDAE Buffer is NULL

*Table 90: Returns for ADAS API **Amba_AdasMDAE_Init**().*

Examples:

```
AMBA_MDAE_CFG_t cfg = {0};
AMBA_MDAE_PAR_t par = {0};

/// set pars
par.Method = MDAE_DEFAULT;
Amba_AdasMDAE_GetDefCfg(&cfg);
rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDAEBuf), cfg.MDAEBuf_Size, 32);
if (rval != OK) {
  AmbaPrint(" Process_MDAE Can't allocate memory.");
  return 0;
}
cfg.RoiData[0].Location.X = 0;
cfg.RoiData[0].Location.Y = 0;
cfg.RoiData[0].Location.W = 12;
cfg.RoiData[0].Location.H = 8;
cfg.RoiData[0].Sensitivity = 2; /// Sensitivity is Multiplier
cfg.RoiData[0].Threshold = 3900;
cfg.RoiData[0].Update_Freq = 1;
cfg.RoiData[0].Update_Cnt = cfg.RoiData[0].Update_Freq - 1;
rval = Amba_AdasMDAE_SetCfg(&cfg);
AmbaPrint("Amba_AdasMDAE_SetCfg  %d\n", rval);
par.IsUpdate = 1;
rval = Amba_AdasMDAE_Init(&par);
```

See Also:

None

2.31. Amba_AdasMDAE_Deinit

Description:

- This function deinitializes motion detection and free buffer.

Parameters:

void

Returns:

void

Examples:

```
AMBA_MDAE_CFG_t cfg = {0};
AMBA_MDAE_PAR_t par = {0};

/// set pars
par.Method = MDAE_DEFAULT;
Amba_AdasMDAE_GetDefCfg(&cfg);
rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDAEBuf), cfg.MDAEBuf_Size, 32);
if (rval != OK) {
  AmbaPrint(" Process_MDAE Can't allocate memory.");
  return 0;
}
cfg.RoiData[0].Location.X = 0;
cfg.RoiData[0].Location.Y = 0;
cfg.RoiData[0].Location.W = 12;
cfg.RoiData[0].Location.H = 8;
cfg.RoiData[0].Sensitivity = 2; /// Sensitivity is Multiplier
cfg.RoiData[0].Threshold = 3900;
cfg.RoiData[0].Update_Freq = 1;
cfg.RoiData[0].Update_Cnt = cfg.RoiData[0].Update_Freq - 1;
rval = Amba_AdasMDAE_SetCfg(&cfg);
AmbaPrint("Amba_AdasMDAE_SetCfg  %d\n", rval);
par.IsUpdate = 1;
rval = Amba_AdasMDAE_Init(&par);
Amba_AdasMDAE_Deinit();
```

See Also:

None

2.32. Amba_AdasMDAE_GetDefCfg

Description:

- This function performs the motion detection process based on Y data.

Parameters:

Type	Parameter	Description
AMBA_MDAE_CFG_t*	**pCfg**	Motion detection configuration. Please refer to Section 2.32.1 for more details.

*Table 91: Parameters for API **Amba_AdasMDAE_GetDefCfg()**.*

Returns:

void.

Examples:

```
AMBA_MDAE_CFG_t cfg = {0};
AMBA_MDAE_PAR_t par = {0};

/// set pars
par.Method = MDAE_DEFAULT;
Amba_AdasMDAE_GetDefCfg(&cfg);
rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDAEBuf), cfg.MDAEBuf_Size, 32);
if (rval != OK) {
  AmbaPrint(" Process_MDAE Can't allocate memory.");
  return 0;
}
cfg.RoiData[0].Location.X = 0;
cfg.RoiData[0].Location.Y = 0;
cfg.RoiData[0].Location.W = 12;
cfg.RoiData[0].Location.H = 8;
cfg.RoiData[0].Sensitivity = 2; /// Sensitivity is Multiplier
cfg.RoiData[0].Threshold = 3900;
cfg.RoiData[0].Update_Freq = 1;
cfg.RoiData[0].Update_Cnt = cfg.RoiData[0].Update_Freq - 1;
rval = Amba_AdasMDAE_SetCfg(&cfg);
AmbaPrint("Amba_AdasMDAE_SetCfg  %d\n", rval);
par.IsUpdate = 1;
rval = Amba_AdasMDAE_Init(&par);
```

See Also:

None

2.32.1.Amba_AdasMDAE_GetDefCfg > AMBA_MDAE_CFG_t

Type	Field	Description
AMBA_MDAE_ROI_DATA_t	**RoiData[MOTION_DETECT _ROI_MAX]**	The region of interest setting. Please refer to Section 2.32.2 for more details.
UINT32	**MDAEBuf_Size**	Buffer size
UINT32	**Lv_Mm_Avg_Size**	MDAE parameters. NOT for setting.
UINT32	**Lv_Snake_Size**	MDAE parameters. NOT for setting.
AMBA_MEM_CTRL_s	**MDAEBuf**	System buffer pointer

*Table 92: Definition of **AMBA_MDAE_CFG_t** for **Amba_AdasMDAE_Proc()**.*

2.32.2.Amba_AdasMDAE_GetDefCfg > AMBA_MDAE_ROI_DATA_t

Type	Field	Description
UINT8	**Valid**	On (1) or OFF(0) this ROI
UINT8	**Sensitivity**	Sensitivity
UINT16	**Tiles_Num**	The tiles number. Please use default settings.
UINT16	**Threshold**	Threshold
UINT16	**Update_Freq**	Data passing frequency
UINT16	**Update_Cnt**	Data passing calculation
AMBA_VA_ROI_s	**Location**	The Region of interest description

*Table 93: Definition of **AMBA_MDAE_ROI_DATA_t** for **Amba_AdasMDAE_Proc**().*

2.33. Amba_AdasMDAE_SetCfg

Description:

- This function sets the configuration of the motion detection process.

Parameters:

Type	Parameter	Description
AMBA_MDAE_CFG_t*	**pCfg**	Motion detection Configuration. Please refer to Section 2.32.1 for more details.

*Table 94: Parameters for API **Amba_AdasMDAE_Init()**.*

Returns:

Return	Description
0	Success
-1	Failure due to parameter errors

*Table 95: Returns for ADAS API **Amba_AdasMDAE_SetCfg()**.*

Examples:

```
AMBA_MDAE_CFG_t cfg = {0};
AMBA_MDAE_PAR_t par = {0};

/// set pars
par.Method = MDAE_DEFAULT;
Amba_AdasMDAE_GetDefCfg(&cfg);
rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.MDAEBuf), cfg.MDAEBuf_Size, 32);
if (rval != OK) {
  AmbaPrint(" Process_MDAE Can't allocate memory.");
  return 0;
}
cfg.RoiData[0].Location.X = 0;
cfg.RoiData[0].Location.Y = 0;
cfg.RoiData[0].Location.W = 12;
cfg.RoiData[0].Location.H = 8;
cfg.RoiData[0].Sensitivity = 2; /// Sensitivity is Multiplier
cfg.RoiData[0].Threshold = 3900;
cfg.RoiData[0].Update_Freq = 1;
cfg.RoiData[0].Update_Cnt = cfg.RoiData[0].Update_Freq - 1;
rval = Amba_AdasMDAE_SetCfg(&cfg);
AmbaPrint("Amba_AdasMDAE_SetCfg  %d\n", rval);
par.IsUpdate = 1;
rval = Amba_AdasMDAE_Init(&par);
```

See Also:

None

3. Quick Start Guide with Example

This section provides some code samples that enables application developers to get started with the ADAS APIs quickly.

3.1. Example: Quick Start ADAS Steps

- **Step 1** – Get default setting and set LDW detection and FCW detection parameters.
  It is assumed that LDWS is to be enabled at speeds greater than 40 km/hr, and Medium sensitivity is desired for left departures while high sensitivity is desired for right departures.

```
AMBA_ADAS_SCENE_PARAMS_s par;
AMBA_ADAS_VIEWANGLE_s Vagl;
AMBA_ADAS_LDW_PARAMS_s ldw_par = {0};
AMBA_ADAS_FCW_PARAMS_s fcw_par = {0};
AMBA_ADAS_LDW_OUTPUTEVENT_s LdwEvent = {0};
AMBA_ADAS_FCW_OUTPUTEVENT_s FcwEvent = {0};
Vagl.HorizAngle = VAUT_HorizAngle;
Vagl.VertAngle = VAUT_VertAngle;
Amba_Adas_GetSceneParams( &par);
par.CameraMountHeight = ADAS_CMTGOLFCLASS ;
par.AutoCalibrationActive = 0;
par.AutoCalibCpuReduction = 0;
par.AutomaticCalibration = 0;
par.HoodLevel = VAUT_HoodLevel;
par.HorizonLevel = VAUT_HorizonLevel;
par.HorizontalPan = 50;
Amba_Adas_SetSceneParams(&par);
```

- **Step 2** – Initialize an ADAS instance
  Assuming that the frame width and height are 576 and 320 respectively, frame format is AMP_YUV_420, horizontal and vertical angle of views are 129 and 83 degrees respectively –

```
Rval = Amba_Adas_Init(img, &par, &Vagl);
```

  If the call was successful, **Amba_Adas_Init()** will return 0. This handle can then be used in all subsequent calls.

- **Step 3** – Set the IsEnable parameters. And, enable the function.
  This can be used run time to arrange the CPU usage.

```
Amba_Adas_GetLdwParams(&ldw_par);
ldw_par.IsEnabled = 1;
Amba_Adas_SetLdwParams(ldw_par);
Amba_Adas_GetFcwParams(&fcw_par);
fcw_par.IsEnabled = 1;
Amba_Adas_SetFcwParams(fcw_par);
```

- **Step 6** – Process frames in a loop

```
while (1)
{
    memset(&gpsInfo, 0, sizeof(gpsInfo));
    gpsInfo.Speed = 100;

    memset(&AuxData, 0, sizeof(AuxData));
    AuxData.pGpsInfo = &gpsInfo;

    OutEvent.FcwEvent = &FcwEvent;
```

```
    OutEvent.LdwEvent = &LdwEvent;
    OutEvent.SceneStatus = &SceneStatus;

    Amba_Adas_Proc( img, &AuxData, &OutEvent, ts);
    if (OutEvent.LdwEvent != NULL) {
        AmbaPrintColor(RED, "Departure: %s \n",(OutEvent.LdwEvent-
>Direction == AMBA_ADAS_DdTowardsLeft) ? "left" : "right");
    }
    if (OutEvent.FcwEvent != NULL) {
        AmbaPrintColor(RED, "Frontal Collision Warning \n");
    }
#if VALOGOUTPUT
    VAAdas_OutputELogger( img, &OutEvent, event, adas_init);
#endif
    if (event % 500 == 0) {
        AmbaPrintColor(RED, "frame number %d \n", event);
    }

}
```

If a LDW event was detected, then p_ldwEvent will be non-NULL. Similarly, if an FCW event was detected, then p_fcwEvent will be non-NULL. p_ldwEvent and p_fcwEvent should never be released.

3.2. Example: Quick Start FCMD Steps

This code sample shows how to use the API functionality of Amba_AdasFCMD_Proc.

- **Step 1** – Get default setting and set parameters

```
AMP_FCMD_CFG_t cfg = {0};
AMP_FCMD_PAR_t par = {0};
Amba_AdasFCMD_GetDef_Cfg_Par(img, &cfg, &par) != OK){
rval = AmbaKAL_MemAllocate(&G_MMPL, &(cfg.Buf), cfg.Buf_Size, 32);

/// set config
cfg.FCMDSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
/// set pars
par.HoodLevel = DEFAULT_HOODLEVEL;
par.HorizonLevel = DEFAULT_HORIZLEVEL;
par.IsUpdate = 1;
rval = Amba_AdasFCMD_SetCfg(&cfg);
```

- **Step 2** – Initialize an FCMD

```
rval = Amba_AdasFCMD_Init(img, &par);
```

If the call was successful, **Amba_AdasFCMD_Init()** will return 0. This handle can then be used in all subsequent calls.

- **Step 3** – Process frames in a loop

```
While (1){
      mwut_gps.Speed = (float)gps_sim[gps_index];
      par.IsGPSandDashboard = 1;
      if (gps_index == 35 ) {
            gps_index = 0;
      }
      rval = Amba_AdasFCMD_Proc(img, &par, &fcmd_event);
      if (rval < 0) {
            fcmd_init = 0;
      }
      if (fcmd_event == AMBA_FCMD_MOVE) {
            AmbaPrintColor(RED, "Frontal Vehicle Move \n");
      }
}
```

3.3. Example: Quick Start LLWS Steps

This code sample shows how to use the API functionality of Amba_AdasFCMD_Proc.

- **Step 1** – Get default setting and set the parameters

```
AMP_LLWS_CFG_t cfg = {0};
AMP_LLWS_PAR_t par = {0};
/// set config
cfg.LLWSSensitivity = ADAS_SL_MEDIUM;
cfg.IsEnabled = 1;
/// set pars
par.HoodLevel = DEFAULT_HOODLEVEL;
par.HorizonLevel = DEFAULT_HORIZLEVEL;
rval = Amba_AdasLLWS_SetCfg(&cfg);
```

- **Step 2** – Initialize a LLWS

```
rval = Amba_AdasLLWS_Init(&par);
```

If the call was successful, **Amba_AdasLLWS_Init()** will return 0. This handle can then be used in all subsequent calls.

- **Step 3** – Process frames in a loop

```
While (1){
      rval = Amba_AdasLLWS_Proc(pdata, &par, &llws_event);
      if (rval < 0) {
            fcmd_init = 0;
      }
      if (llws_event == AMBA_LLWS_LOW_LIGHT) {
            AmbaPrintColor(RED, "Turn on the light \n");

      }
}
```

3.4. Example: Print Calibration Levels in Manual Mode

This code sample shows how to use the API functionality of **Amba_Adas_GetSceneStatus()** to print the current horizon and hood calibration levels. Note that **Amba_Adas_GetSceneStatus()** will provide current horizon and hood levels **in manual calibration mode only**. This functionality can be used for debugging or for visual overlays.

```
AMBA_ADAS_SCENE_PARAMS_s params;
Amba_Adas_GetSceneParams( &params);
printf("horizon_level=%d, hood_level=%d\n", (int)params.HorizonLevel,
(int)params.HoodLevel);
```