H.264 Encoder v01.00 to H.264 Encoder v02.00 on HDVICP2 based SOC

Migration Guide

Template Revision History

Rev No	Author(s)	Revision History	Date	Approval(s)
0.1	Sayanna, Santoshkumar S K	Initial draft	Apr 20, 2012	

TABLE OF CONTENTS

1 Introduction

This document describes the considerations for migrating from H264 Encoder Product 1.0 to H264 Encoder Product 2.0.The document provides the details of modified and newly introduced interface parameters.

Being compared to H264 Encoder Product 1.0, H264 Encoder Product 2.0 supports many additional features such as

- **Gradual Decoder Refresh (GDR)**
- **High Mega pixel Encoding**
- **Region of Interest Encoding (ROI)**
- **Privacy Mask**
- **Dynamic change of rate control algorithm, intra refresh**
- **Rate Control, Control for max and min Pic Size Ratio for each picture type**
- **Water Marking**
- **Encoding N Frames per process call**
- **H.264 Lite (High Speed) configuration for higher performance**

At the same time few feature enhancements like below are added

- **Long Term Reference Picture (LTRP) mechanism is enhanced**
- **Rate Control improvements such as HRD compliant Rate control mode, CVBR Rate control and continuous skip frame prevention at Low bit Rate are added.**

Note that since this document describes the migration from H264 Encoder Product 1.0 to H264 Encoder Product 2.0, familiarity with H264 Encoder Product 1.0 is assumed.

The whole document shall have the naming convention **Product 1.0** for H264 Encoder v01.00.04.01 and **Product 2.0** for H264 Encoder v02.00.07.01

2 Create Parameters

This structure defines the Create time parameters for all H264ENC objects.

Product 1.0	Product 2.0
typedef struct	typedef struct
{	{
IVIDENC2_Params videnc2Params ;	IVIDENC2_Params videnc2Params ;
IH264ENC_RateControlParams rateControlParams ;	IH264ENC_RateControlParams rateControlParams ;
IH264ENC_InterCodingParams interCodingParams ;	IH264ENC_InterCodingParams interCodingParams ;
IH264ENC_IntraCodingParams intraCodingParams ;	IH264ENC_IntraCodingParams intraCodingParams ;
IH264ENC_NALUControlParams nalUnitControlParams;	IH264ENC_NALUControlParams nalUnitControlParams ;
IH264ENC_SliceCodingParams sliceCodingParams ;	IH264ENC_SliceCodingParams sliceCodingParams ;
IH264ENC_LoopFilterParams loopFilterParams ;	IH264ENC_LoopFilterParams loopFilterParams ;
IH264ENC_FMOCodingParams fmoCodingParams ;	IH264ENC_FMOCodingParams fmoCodingParams ;
IH264ENC_VUICodingParams vuiCodingParams ;	IH264ENC_VUICodingParams vuiCodingParams ;
IH264ENC_StereoInfoParams stereoInfoParams ;	IH264ENC_StereoInfoParams stereoInfoParams ;
IH264ENC_FramePackingSEIParams framePackingSEIParams ;	IH264ENC_FramePackingSEIParams framePackingSEIParams ;
	IH264ENC_SVCCodingParams svcCodingParams ;
XDAS_Int8 interlaceCodingType ;	XDAS_Int8 interlaceCodingType ;
XDAS_Int8 bottomFieldIntra ;	XDAS_Int8 bottomFieldIntra ;
XDAS_Int8 gopStructure ;	XDAS_Int8 gopStructure ;
XDAS_Int8 entropyCodingMode ;	XDAS_Int8 entropyCodingMode ;
XDAS_Int8 transformBlockSize ;	XDAS_Int8 transformBlockSize ;
XDAS_Int8 log2MaxFNumMinus4 ;	XDAS_Int8 log2MaxFNumMinus4 ;
XDAS_Int8 picOrderCountType ;	XDAS_Int8 picOrderCountType ;
	XDAS_Int8 enableWatermark ;
XDAS_Int32 IDRFrameInterval ;	XDAS_Int32 IDRFrameInterval ;
XDAS_Int32 pConstantMemory ;	XDAS_Int32 pConstantMemory ;
XDAS_Int32 maxIntraFrameInterval ;	XDAS_Int32 maxIntraFrameInterval ;
XDAS_UInt32 debugTraceLevel ;	XDAS_UInt32 debugTraceLevel ;
XDAS_UInt32 lastNFramesToLog ;	XDAS_UInt32 lastNFramesToLog ;
XDAS_Int8 enableAnalyticinfo ;	XDAS_Int8 enableAnalyticinfo ;
XDAS_Int8 enableGMVSei ;	XDAS_Int8 enableGMVSei ;
XDAS_Int8 constraintSetFlags ;	XDAS_Int8 constraintSetFlags ;
XDAS_Int8 enableRCDO ;	XDAS_Int8 enableRCDO ;
XDAS_Int32 enableLongTermRefFrame ;	XDAS_Int32 enableLongTermRefFrame ;
	XDAS_Int32 LTRPPeriod ;
XDAS_Int8 numTemporalLayer ;	XDAS_Int8 numTemporalLayer ;
XDAS_Int8 referencePicMarking ;	XDAS_Int8 referencePicMarking ;
XDAS_Int32 reservedParams[3] ;	XDAS_Int32 reservedParams[3] ;
} IH264ENC_Params;	} IH264ENC_Params;

Below is the description of the newly added parameters

2.1 IH264ENC_SVCCodingParams svcCodingParams

This structure Controls the SVC coding parameters. Refer Annex G of the H.264 standard for more details of SVC and parameters. Though it is supported in later version of product 1.0 line i.e., v01.00.03.00 onwards, but kept it here for the users migrating from versions previous to this. The structure definition of IH264ENC_SVCCodingParams is as given below,

```
typedef struct IH264ENC_SVCCodingParams
{
        XDAS_UInt8          svcExtensionFlag;
        XDAS_UInt8          dependencyID;
        XDAS_UInt8          qualityID;
        XDAS_UInt8          enhancementProfileID;
        XDAS_UInt8          layerIndex;
        XDAS_Int8          refLayerDQId;
} IH264ENC_SVCCodingParams;
```

2.1.1 svcExtensionFlag

This parameter when enabled, configures the codec to put SVC extensions in the bit-stream. For normal H.264 operation this Flag needs to be 0 (default value is IH264_SVC_EXTENSION_FLAG_DISABLE). For Encoder instance to encode SSPS, Prefix-NALU, Coded Slice in the bit-stream, this flag needs to be set. The enumeration of the possible values are,

```
typedef enum
{
        /*Svc Extension Flag Disabled*/
         IH264_SVC_EXTENSION_FLAG_DISABLE          = 0 ,
         /*Svc Extension Flag Enabled*/
         IH264_SVC_EXTENSION_FLAG_ENABLE           = 1 ,
       /*Svc Extension Flag Enabled with EC Flexibility*/
         IH264_SVC_EXTENSION_FLAG_ENABLE_WITH_EC_FLEXIBILITY = 2
} IH264ENC_SvcExtensionFlag;
```

IH264_SVC_EXTENSION_FLAG_DISABLE: Disables all SVC features/syntaxes and rest of the structure is not read/respected.
IH264_SVC_EXTENSION_FLAG_ENABLE: Encodes the required SVC related syntaxes of the layer for which H.264 Codec has been instantiated.
IH264_SVC_EXTENSION_FLAG_ENABLE_WITH_EC_FLEXIBILITY: Encodes the required SVC related syntaxes of the layer for which H.264 Codec has been instantiated. This mode is used to generate the bitstream which is compatible to TI-SVC decoder and which will make work easy of TI-SVC decoder's Error Concelment by putting info no_inter_layer_pred_flag in the svc-bitstream slice header.

Note that if svcExtensionFlag is IH264_SVC_EXTENSION_FLAG_DISABLE then rest all parameters of this structure IH264ENC_SVCCodingParams are ignored.

2.1.2 dependencyID

This parameter tells whether the current instance is for Base layer or for enhancement layer and also conveys Layer ID Info. If the instance has to be in Base layer then its value would be BASE_LAYER (0).

2.1.3 qualityID

This parameter tells Quality ID of the layer that the current instance of encoder is going to encode. For Base Layer its value would be BASE_LAYER (0).

2.1.4 enhancementProfileID

This parameter conveys the enhancement encoder instance like what should be the profile ID to be encoded in the Sub-Sequence Parameter Set (SSPS). Possible values are IH264SVC_BASELINE_PROFILE (83) or IH264SVC_HIGH_PROFILE (86).

2.1.5 layerIndex

This parameter conveys the enhancement encoder instance like what should be the pic_parameter_set_id and seq_parameter_set_id to be encoded in the Picture Parameter Set (PPS) and Sub-Sequence Parameter Set (SSPS).

2.1.6 refLayerDQId

This parameter conveys the DQ Id of the Reference Layer.

2.2 enableWatermark

This Parameter Enables or disables Water Mark SEI message in the bit stream. Watermarking is a mechanism to add identity to a bit stream to help decoder to identify the media content. For video security applications, it has become a de-facto requirement to prevent tampering with video. HDVICP2 H.264 codecs should support a watermarking scheme at no loss in performance. The possible values for this parameter are zero (disable) and non-zero (enable). For more details, see Appendix L in H.264 Encoder 2.0 User's Guide.

2.3 LTRPPeriod

This parameter is used to specify the long-term reference frame marking interval. This parameter is in use when enableLongTermRefFrame is IH264ENC_LTRP_REFERTOP_REACTIVE(3)or IH264ENC_LTRP_REFERTO_PERIODICLTRP (1). When Hierarchical coding is enabled this value should be multiple of subGop length.

3 Dynamic Parameters

This structure defines the run time parameters for all H264ENC objects.

Product 1.0	Product 2.0
typedef struct IH264ENC_DynamicParams { IVIDENC2_DynamicParams videnc2DynamicParams; IH264ENC_RateControlParams rateControlParams ; IH264ENC_InterCodingParams interCodingParams ; IH264ENC_SliceCodingParams sliceCodingParams ; XDAS_Int32 sliceGroupChangeCycle ; XDM_Point searchCenter ; XDAS_Int8 enableStaticMBCount ; XDAS_Int32 reservedDynParams[4] ; } IH264ENC_DynamicParams;	typedef struct IH264ENC_DynamicParams { IVIDENC2_DynamicParams videnc2DynamicParams; IH264ENC_RateControlParams rateControlParams ; IH264ENC_InterCodingParams interCodingParams ; **IH264ENC_IntraCodingParams intraCodingParams ;** IH264ENC_SliceCodingParams sliceCodingParams ; XDAS_Int32 sliceGroupChangeCycle ; XDM_Point searchCenter ; XDAS_Int8 enableStaticMBCount ; **XDAS_Int32 enableROI;** XDAS_Int32 reservedDynParams[1]; } IH264ENC_DynamicParams;

Below is the description of the newly added parameters

3.1 IH264ENC_IntraCodingParams intraCodingParams

In Product 1.0 H264 Encoder these parameters were only a part of create time parameters but in Product 2.0 Encoder they became a part of Dynamic parameters too. These are included in dynamic parameters to have a flexibility of changing the intra coding parameters dynamically viz., to switch from Normal Intra coding mode to High Speed Intra coding mode. The difference of IH264ENC_IntraCodingParams structure between Product 1.0 and Product 2.0 is as shown in a below table,

Product 1.0	Product 2.0
typedef struct IH264ENC_IntraCodingParams { XDAS_Int8 intraCodingPreset ; XDAS_Int16 lumaIntra4x4Enable ; XDAS_Int16 lumaIntra8x8Enable ; XDAS_Int8 lumaIntra16x16Enable ; XDAS_Int8 chromaIntra8x8Enable ; XDAS_Int8 chromaComponentEnable ; XDAS_Int8 intraRefreshMethod ; XDAS_Int16 intraRefreshRate ; XDAS_Int16 constrainedIntraPredEnable ; } IH264ENC_IntraCodingParams ;	typedef struct IH264ENC_IntraCodingParams { XDAS_Int8 intraCodingPreset ; XDAS_Int16 lumaIntra4x4Enable ; XDAS_Int16 lumaIntra8x8Enable ; XDAS_Int8 lumaIntra16x16Enable ; XDAS_Int8 chromaIntra8x8Enable ; XDAS_Int8 chromaComponentEnable ; XDAS_Int8 intraRefreshMethod ; XDAS_Int16 intraRefreshRate ; **XDAS_Int16 gdrOverlapRowsBtwFrames ;** XDAS_Int16 constrainedIntraPredEnable ; **XDAS_Int8 intraCodingBias ;** } IH264ENC_IntraCodingParams ;

3.1.1 intraCodingPreset

H264 Product 2.0 Encoder has a feature of HIGH_SPEED encoding. The intra coding preset in this mode would be the below added presets.

Product 1.0	Product 2.0
typedef enum { IH264_INTRACODING_DEFAULT = 0 , /**< Default intra coding params */ IH264_INTRACODING_USERDEFINED = 1 , /**< User defined intra coding params */ IH264_INTRACODING_MAX } IH264ENC_IntraCodingPreset;	typedef enum { IH264_INTRACODING_DEFAULT = 0 , /**< Default intra coding params */ IH264_INTRACODING_USERDEFINED = 1 , /**< User defined intra coding params */ **IH264_INTRACODING_EXISTING = 2 ,** **IH264_INTRACODING_HIGH_SPEED = 3 , /**< High Speed intra Coding Preset */** IH264_INTRACODING_MAX } IH264ENC_IntraCodingPreset;

3.1.2 *intraRefreshMethod*

One more intra refresh method being added in H264 Product 2.0 Encoder is IH264_INTRAREFRESH_GDR. For detailed description of GDR feature, see Appendix O in H.264 Encoder 2.0 User's Guide.

Product 1.0	Product 2.0
typedef enum { IH264_INTRAREFRESH_NONE = 0 , /**< Doesn't insert forcefully intra macro blocks */ IH264_INTRAREFRESH_DEFAULT = IH264_INTRAREFRESH_NONE, /**< Default intra refresh is OFF */ IH264_INTRAREFRESH_CYCLIC_MBS , /**< Insters intra macro blocks in a * cyclic fashion cyclic interval is * equal to intraRefreshRate */ IH264_INTRAREFRESH_CYCLIC_SLICES , /**< Insters Intra Slices(Row based) in * a cyclic fashion: * cyclic interval is equal to * intraRefreshRate */ IH264_INTRAREFRESH_RDOPT_MBS , /**< position of intra macro blocks is * intelligently chosen by encoder, * but the number of forcely coded * intra macro blocks in a frame is * gaurnteed to be equal to * totalMbsInFrame/intraRefreshRate */ IH264_INTRAREFRESH_MAX } IH264ENC_IntraRefreshMethods ;	typedef enum { IH264_INTRAREFRESH_NONE = 0 , /**< Doesn't insert forcefully intra macro blocks */ IH264_INTRAREFRESH_DEFAULT = IH264_INTRAREFRESH_NONE, /**< Default intra refresh is OFF */ IH264_INTRAREFRESH_CYCLIC_MBS , /**< Insters intra macro blocks in a * cyclic fashion cyclic interval is * equal to intraRefreshRate */ IH264_INTRAREFRESH_CYCLIC_SLICES , /**< Insters Intra Slices(Row based) in * a cyclic fashion: * cyclic interval is equal to * intraRefreshRate */ IH264_INTRAREFRESH_RDOPT_MBS , /**< position of intra macro blocks is * intelligently chosen by encoder, * but the number of forcely coded * intra macro blocks in a frame is * gaurnteed to be equal to * totalMbsInFrame/intraRefreshRate */ **IH264_INTRAREFRESH_GDR , /**< Instead of a sudden Intra Refresh** *** of entire frame, the frame is refreshed** *** Gradualy over a duration (which is con-** *** figerable) of frames with refresh** *** happening by Intra coded rows scanning** ***from top to bottom of the scene/picture.** ***/** IH264_INTRAREFRESH_MAX } IH264ENC_IntraRefreshMethods ;

3.1.3 *gdrOverlapRowsBtwFrames*

Number of GDR'd rows overlap between successive GDR frames. Note that gdrOverlapRowsBtwFrames should be less than intraRefreshRate.

3.1.4 *intraCodingBias*

This parameter is added to have an option of enabling Normal/High speed intra coding. Enumerations for this parameter are,

```
{
        IH264ENC_INTRACODINGBIAS_NORMAL     = 0 ,
        /**< Normal number of intra Mbs        */
        IH264ENC_INTRACODINGBIAS_HIGH_SPEED  = 12 ,
       /**< intra Mbs restricted for HIGH SPEED */
        IH264ENC_INTRACODINGBIAS_DEFAULT=
        IH264ENC_INTRACODINGBIAS_NORMAL ,
      /**< Default intra codign bias         */
        IH264ENC_INTRACODINGBIAS_MAX
} IH264ENC_IntraCodingBias;
```

3.2 enableROI

H264 Product 2.0 Encoder supports Region-of-interest (ROI) based coding which would allow the encoder to provide more/less importance to a subset of input image data. So this dynamic parameter enables/disables the ROI coding for a particular frame. The possible values of this parameter are zero and non-zero. For more details on ROI feature, see Appendix K in H.264 Encoder 2.0 User's Guide.

4 InArgs and OutArgs

4.1 InArgs

This structure defines the input arguments being passed to H.264 encoder.

Product 1.0	Product 2.0
typedef struct IH264ENC_InArgs { IVIDENC2_InArgs videnc2InArgs; } IH264ENC_InArgs;	typedef struct IH264ENC_InArgs { IVIDENC2_InArgs videnc2InArgs ; **XDAS_Int32 processId;** **IH264ENC_RoiInput roiInputParams;** **XDAS_UInt32 inputKey ;** } IH264ENC_InArgs;

Below is the description of the newly added parameters

4.1.1 processId

The processId in InArgs was kept to ease the acquire time optimization in application code. In N channel case, acquire is happening for last channel and this (processId) as argument is passed into acquire call. This will make application to understand that for which process call, acquire has been made. With this information application can optimize the time spent in acquire. Like, it might have happened that from last call of acquire, HDVICP2 became unavailable to any further process call(s). In this scenario application will get to know that HDVICP2 was not given to somebody else from last process call, and hence it can do some optimization in acquire routine.

4.1.2 IH264ENC_RoiInput roiInputParams

This structure defines the ROI input parameters required by Encoder. These parameters would be effective only if the dynamic parameter enableROI is enabled (!0). The structure definition of IH264ENC_RoiInput is as below,

```
typedef struct IH264ENC_RoiInput
{
        XDM_Rect   listROI[IH264ENC_MAX_ROI];
        XDAS_Int8  roiType[IH264ENC_MAX_ROI];
        XDAS_Int8  numOfROI;
        XDAS_Int32 roiPriority[IH264ENC_MAX_ROI];
}IH264ENC_RoiInput;
```

The value of **IH264ENC_MAX_ROI** is 36, it means that the current design supports maximum ROI's up to 36.

4.1.2.1 listROI

This buffer holds the list of ROI's with their x and y co-ordinates.

4.1.2.2 roiType

This buffer holds the type of each ROI. The enumeration IH264ENC_RoiType defines the different ROI types

```
typedef enum
{
        IH264_FACE_OBJECT      = 0,
        /**< Face type of ROI object            */
        IH264_BACKGROUND_OBJECT = 1,
        /**< Background type of ROI object      */
        IH264_FOREGROUND_OBJECT = 2,
```

```
        /**< Foreground type of ROI object        */
        IH264_DEFAULT_OBJECT    = 3,
        /**< Default type of ROI object           */
        IH264_PRIVACY_MASK      = 4
        /**< Privacy mask type of ROI object      */
    } IH264ENC_RoiType;
```

4.1.2.3 numOfROI

This parameter contains the number of ROIs passed to codec.

4.1.2.4 roiPriority

This buffer contains the priority information of each ROI.

4.1.3 inputKey

This parameter along with the few important properties of a frame is used to generate the Encrypted key. If watermarking is enabled then this Encrypted key would be inserted in the form of user data unregistered SEI message in the encoded stream.

4.2 OutArgs

This structure defines the output argument being generated from H.264 encoder.

The changes between product 1.0 and product 2.0 w.r.t this structure are not major but only parameter extErrorCode[] has been added to pass sub extended errors from codec to application/user. This parameter will hold the sub-extended error(s) generated due to fail in certain parameter setting. Reading this parameter, user can adjust his configuration of encoder.

Product 1.0	Product 2.0
typedef struct IH264ENC_OutArgs { IVIDENC2_OutArgs videnc2OutArgs ; XDAS_Int32 bytesGeneratedBotField ; XDAS_Int32 vbvBufferLevel ; XDAS_Int32 numStaticMBs ; XDAS_Int32 control ; } IH264ENC_OutArgs;	typedef struct IH264ENC_OutArgs { IVIDENC2_OutArgs videnc2OutArgs ; XDAS_Int32 bytesGeneratedBotField ; XDAS_Int32 vbvBufferLevel ; XDAS_Int32 numStaticMBs ; **XDAS_Int32 temporalId ;** XDAS_Int32 control ; **XDAS_UInt32 extErrorCode[IH264ENC_EXTERROR_NUM_MAXWORDS];** } IH264ENC_OutArgs;

4.2.1 temporalId

This parameter carries the temporal layer Id of current frame in Hierarchical encoding. If `IH264ENC_SVCCodingParams::svcExtensionFlag` is enabled then the bit-stream shall have prefix-NALU. The temporal_id value in the prefix-NALU and the value of this parameter are same. Instead of decoding the prefix-NALU, the user can easily read the value of the current frame's temporal id from this parameter.

4.2.2 extErrorCode

Parameter extErrorCode[] has been added to pass sub extended errors from codec to application/user. This parameter will hold the sub-extended error(s) generated due to fail in certain parameter setting. Reading this parameter, user can adjust his configuration of encoder. This parameter shall also contain the information of run time encoder fail.

5 Enhancements and Miscellaneous Changes

Some of the features like long term reference picture (LTRP) mechanism, Rate Control mechanism are enhanced in the Product 2.0 H264 Encoder. This encoder has got the feature to encode the frame of all the fed channels into a single process call which is called as Multi-Channel (N-Channel) Frame processing Encoder. **PERIODICLTRP**

5.1 Long Term Reference Pictures (LTRP)

There are 3 schemes of LTRP. Enumerations and a brief definition of the enhanced schemes are,

Product 1.0	Product 2.0
typedef enum { IH264ENC_LTRP_NONE = 0, IH264ENC_LTRP_REFERTO_REFERTOIDR = 1, IH264ENC_LTRP_REFERTOP_PROACTIVE =2, IH264ENC_LTRP_REFERTOP_REACTIVE = 3 }IH264ENC_LTRPScheme;	typedef enum { IH264ENC_LTRP_NONE = 0, IH264ENC_LTRP_REFERTO_ **PERIODICLTRP** = 1, IH264ENC_LTRP_REFERTOP_PROACTIVE =2, IH264ENC_LTRP_REFERTOP_REACTIVE = 3 }IH264ENC_LTRPScheme;

LTRP option 1 is IH264ENC_LTRP_REFERTO**IDR** in product 1.0 and IH264ENC_LTRP_REFERTO_**PERIODICLTRP** in product 2.0.

Description for Product 1.0 H264 Encoder: Mark all I frames as long term-reference frames and based on the frame control IH264ENC_Control, refer to a long-term reference frame (I frame).

Description for Product 2.0 H264 Encoder: Mark frames as long-term reference frame with the period given by LTRPPeriod of IH264ENC_Params and based on the frame control IH264ENC_Control, refer to the long-term reference frame.

IH264ENC_LTRP_REFERTOP_REACTIVE: This option is not supported in product 1.0 but supported in product 2.0. Mark frames as long-term reference frame with the period given by LTRPPeriod of IH264ENC_Params. At any point of time there will be 2 long-term frames and based on the frame control IH264ENC_Control, refer to the last long-term reference frame. For more details, see Appendix H in H.264 Encoder 2.0 User's Guide.

5.2 Rate Control Improvements

As compared to product 1.0, we have very good additional/enhanced features in product 2.0 Rate control. Features are Dynamic change of rate control algorithm, Control for max and min Pic Size Ratio for each picture type, Rate Control improvements such as HRD compliant Rate control mode, CVBR Rate control and continuous skip frame prevention at Low bit Rate. For more details on CVBR, see Appendix N in H.264 Encoder 2.0 User's Guide.
To implement the above mentioned features, few parameters in the rateControl structure are added as shown.

Product 1.0	Product 2.0
typedef struct IH264ENC_RateControlParams {	typedef struct IH264ENC_RateControlParams {
XDAS_Int8 rateControlParamsPreset ;	XDAS_Int8 rateControlParamsPreset ;
XDAS_Int8 scalingMatrixPreset ;	XDAS_Int8 scalingMatrixPreset ;
XDAS_Int8 rcAlgo ;	XDAS_Int8 rcAlgo ;
XDAS_Int8 qpI ;	XDAS_Int8 qpI ;
XDAS_Int8 qpMaxI ;	XDAS_Int8 qpMaxI ;
XDAS_Int8 qpMinI ;	XDAS_Int8 qpMinI ;
XDAS_Int8 qpP ;	XDAS_Int8 qpP ;
XDAS_Int8 qpMaxP ;	XDAS_Int8 qpMaxP ;
XDAS_Int8 qpMinP ;	XDAS_Int8 qpMinP ;
XDAS_Int8 qpOffsetB ;	XDAS_Int8 qpOffsetB ;
XDAS_Int8 qpMaxB ;	XDAS_Int8 qpMaxB ;
XDAS_Int8 qpMinB ;	XDAS_Int8 qpMinB ;
XDAS_Int8 allowFrameSkip ;	XDAS_Int8 allowFrameSkip ;
XDAS_Int8 removeExpensiveCoeff ;	XDAS_Int8 removeExpensiveCoeff ;
XDAS_Int8 chromaQPIndexOffset ;	XDAS_Int8 chromaQPIndexOffset ;
XDAS_Int8 IPQualityFactor ;	XDAS_Int8 IPQualityFactor ;
XDAS_Int32 initialBufferLevel ;	XDAS_Int32 initialBufferLevel ;
XDAS_Int32 HRDBufferSize ;	XDAS_Int32 HRDBufferSize ;
XDAS_Int16 minPicSizeRatio ;	XDAS_Int16 minPicSizeRatio**I** ;
XDAS_Int16 maxPicSizeRatio ;	XDAS_Int16 maxPicSizeRatio**I** ;
	XDAS_Int16 minPicSizeRatioP ;
	XDAS_Int16 maxPicSizeRatioP ;
	XDAS_Int16 minPicSizeRatioB ;
	XDAS_Int16 maxPicSizeRatioB ;
XDAS_Int8 enablePRC ;	XDAS_Int8 enablePRC ;
XDAS_Int8 enablePartialFrameSkip ;	XDAS_Int8 enablePartialFrameSkip ;
XDAS_Int16 reserved ;	XDAS_Int8 reserved ;
	XDAS_Int32 VBRDuration ;
	XDAS_Int8 VBRsensitivity ;
	XDAS_Int16 skipDistributionWindowLength;
	XDAS_Int16 numSkipInDistributionWindow;
	XDAS_Int8 enableHRDComplianceMode ;
	XDAS_Int32 frameSkipThMulQ5 ;
	XDAS_Int32 vbvUseLevelThQ5 ;
	XDAS_Int32 reservedRC[3] ;
XDAS_Int32 reservedRC[3] ;	} IH264ENC_RateControlParams ;
} IH264ENC_RateControlParams ;	

Description of newly added parameters

5.2.1 minPicSizeRatioI

This ratio is used to compute minimum I picture size and the allowed values are from 1 to 4. Setting this to 0 will enable the encoder on its own to chose the ratio. Note that this is guided value to rate control to determine min picture size and encoder may not strictly follow this. The user has to specify this value in Q5 format.
Minimum picture size is computed in the following manner,
If minPicSizeRatioI is from 1 to 4, minPicSize = averagePicSize >> minPicSizeRatioI
If minPicSizeRatioI is from 5 to 31 minPicSize = averagePicSize * minPicSizeRatioI >> Q5

5.2.2 maxPicSizeRatioI

This ratio is used to compute maximum I picture size and the allowed values are from 2 to 30 and 33 to 960. Setting this to 0 and 1 will enable the encoder to choose the ratio on its own. Note that this is guided value to rate control to determine max picture size and encoder may not strictly follow this. The user has to specify this value in Q5 format.
Maximum picture size is computed in the following manner,
If maxPicSizeRatioI is from 2 to 30 maxPicSize = averagePicSize * maxPicSizeRatioI
If maxPicSizeRatioI is from 33 to 960 maxPicSize = averagePicSize * maxPicSizeRatioI >> Q5

The above description holds good for (**minPicSizeRatioP**, **maxPicSizeRatioP**) and (**minPicSizeRatioB**, **maxPicSizeRatioB**) respectively for P and B frames. These parameters are added to Control the max and min Pic Size Ratio for each picture type.

5.2.3 VBRDuration

Duration over which statistics are collected to switch bit-rate states.Increasing this value will make VBR wait for longer time before switching bit-rate state.

5.2.4 VBRsensitivity

Specifies the target bitrate used by rate control in high complexity state.

5.2.5 skipDistributionWindowLength

This parameter contains the number of frames over which the skip frames can be distributed.

5.2.6 numSkipInDistributionWindow

This parameter contains the number of skips allowed within the distribution window.

5.2.7 enableHRDComplianceMode

This parameter allows encoder to generate HRD compliant bit-streams. Allowed values are 0 and 1.

5.2.8 frameSkipThMulQ5

This parameter is added for CBR quality improvement

5.2.9 vbvUseLevelThQ5

This parameter is added for CBR quality improvement

5.3 N Channel Processing Feature support

This feature supports encoding N frames processing in a single process call. Here N being the number of channel entries in a process call. The interface changes to support this feature are as below,

IH264ENC_Fxns

Product 1.0	Product 2.0
typedef struct IH264ENC_Fxns { IVIDENC2_Fxns ividenc; } IH264ENC_Fxns;	typedef struct IH264ENC_Fxns { IVIDENC2_Fxns ividenc; **XDAS_Int32 (*processMulti) (IH264ENC_ProcessParamsList *processList);** } IH264ENC_Fxns;

Function pointer, **(*processMulti) (IH264ENC_ProcessParamsList *processList)**: This function pointer is added to support frame processing of all the fed channels in a single process call. The structure **IH264ENC_ProcessParamsList** is declared as below,

```
typedef struct
{
        XDAS_Int32 numEntries ;
        XDAS_Int32 enableErrorCheck  ;
        IH264ENC_ProcessParams
processParams[IH264ENC_MAX_LENGTH_PROCESS_LIST];
} IH264ENC_ProcessParamsList;
```

5.3.1 numEntries

This parameter contains the number of channels to be encoded in the process call.

5.3.2 enableErrorCheck

This parameter controls (enables/disables) the check of the non supported features in N channel scenario.

Note:

1. Effect of flag is as of now only if (numEntries > 1).
2. Suggested value for this flag is 0 to have better performance

5.3.3 IH264ENC_ProcessParams processParams

This structure defines the channel specific parameters passed in a single process call. An array holds the process parameters viz., handle,InArgs,outArgs etc of the channel(s) to be processed. The IH264ENC_ProcessParams structure declaration is as follows,

```
typedef struct
{
        IVIDENC2_Handle handle;
        IVIDEO2_BufDesc *inBufs;
        XDM2_BufDesc *outBufs;
        IVIDENC2_InArgs *inArgs;
        IVIDENC2_OutArgs *outArgs;
} IH264ENC_ProcessParams;
```

***handle*:** Handle to an algorithm instance
inBufs: pointer to detailed buffer descriptor for video buffers
outBufs: pointer to the Buffer descriptors

inArgs: pointer to the input argument being passed to encoder
outArgs: pointer to the output argument being generated from encoder

For more details on this feature, see Appendix M in H.264 Encoder 2.0 User's guide.

5.4 Miscellaneous Structures

H264 Product 2.0 Encoder has a feature of HIGH_SPEED Encoding. In this feature, we shall have a defined mode of inter and intra coding methods. The changes related to the intra coding structure are already described in the Create parameters section.

IH264ENC_InterCodingParams: This structure is a part of create parameters and Dynamic parameters.

Product 1.0	Product 2.0
typedef struct IH264ENC_InterCodingParams {	typedef struct IH264ENC_InterCodingParams {
XDAS_Int8 interCodingPreset ;	XDAS_Int8 interCodingPreset ;
XDAS_Int16 searchRangeHorP ;	XDAS_Int16 searchRangeHorP ;
XDAS_Int16 searchRangeVerP ;	XDAS_Int16 searchRangeVerP ;
XDAS_Int16 searchRangeHorB ;	XDAS_Int16 searchRangeHorB ;
XDAS_Int16 searchRangeVerB ;	XDAS_Int16 searchRangeVerB ;
XDAS_Int8 interCodingBias ;	XDAS_Int8 interCodingBias ;
XDAS_Int8 skipMVCodingBias ;	XDAS_Int8 skipMVCodingBias ;
XDAS_Int8 minBlockSizeP ;	XDAS_Int8 minBlockSizeP ;
XDAS_Int8 minBlockSizeB ;	XDAS_Int8 minBlockSizeB ;
	XDAS_Int8 meAlgoMode ;
} IH264ENC_InterCodingParams ;	} IH264ENC_InterCodingParams ;

interCodingPreset: For the existing inter coding presets, 2 more being to define High speed inter coding and Medium speed_High Quality inter coding.

Product 1.0	Product 2.0
typedef enum	typedef enum
{	{
IH264_INTERCODING_DEFAULT = 0 , /**< Default Inter coding params */	IH264_INTERCODING_DEFAULT = 0 , /**< Default Inter coding params */
IH264_INTERCODING_USERDEFINED = 1 , /**< User defined inter coding params */	IH264_INTERCODING_USERDEFINED = 1 , /**< User defined inter coding params */
IH264_INTERCODING_EXISTING = 2 , /**< Keep the inter coding params as * existing. This is useful because * during control call if user don't * want to chnage the inter coding Params */	IH264_INTERCODING_EXISTING = 2 , /**< Keep the inter coding params as * existing. This is useful because * during control call if user don't * want to chnage the inter coding Params */
	IH264_INTERCODING_MED_SPEED_HIGH_QUALITY = 3 , /< Med Speed High Quality*/**
	IH264_INTERCODING_HIGH_SPEED = 4, /< High Speed Preset*/**
IH264_INTERCODING_MAX	IH264_INTERCODING_MAX
} IH264ENC_InterCodingPreset;	

	} IH264ENC_InterCodingPreset;

meAlgoMode: This parameter controls the ME algorithm mode. The enumerations for this parameter are,

```
typedef enum
{
        IH264ENC_MOTIONESTMODE_NORMAL     = 0 ,
        /**< Normal meAlgo        */
        IH264ENC_MOTIONESTMODE_HIGH_SPEED  = 1 ,
        /**< meAlgo for HIGH SPEED */
        IH264ENC_MOTIONESTMODE_DEFAULT                    =
        IH264ENC_MOTIONESTMODE_NORMAL ,
    /**< Default meAlgo       */
        IH264ENC_MOTIONESTMODE_MAX
} IH264ENC_MeAlgoMode;
```

Control call for GDR: One more option in the control call is being added to enable Gradual Decoder Refresh (GDR) feature.

Product 1.0	Product 2.0
IH264ENC_CTRL_REFER_LONG_TERM_FRAME = XDM_CUSTOMENUMBASE, /**< Refere long term reference frame (I/IDR frames) when * IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOIDR */ IH264ENC_CTRL_NOWRITE_NOREFUPDATE, /**< Current frame is a non-referencing P frame and do * not update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_WRITE_NOREFUPDATE, /**< Current frame is a referencing P frame and do * not update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_NOWRITE_REFUPDATE, /**< Current frame is a non-referencing P frame and * update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_WRITE_REFUPDATE, /**< Current frame is a referencing P frame and * update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ } IH264ENC_Control;	IH264ENC_CTRL_REFER_LONG_TERM_FRAME = XDM_CUSTOMENUMBASE, /**< Refere long term reference frame (I/IDR frames) when * IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOIDR */ IH264ENC_CTRL_NOWRITE_NOREFUPDATE, /**< Current frame is a non-referencing P frame and do * not update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_WRITE_NOREFUPDATE, /**< Current frame is a referencing P frame and do * not update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_NOWRITE_REFUPDATE, /**< Current frame is a non-referencing P frame and * update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ IH264ENC_CTRL_WRITE_REFUPDATE, /**< Current frame is a referencing P frame and * update the reference frame for this frame. Applicable * when IH264ENC_LTRPScheme is IH264ENC_LTRP_REFERTOP_PROACTIVE */ **IH264ENC_CTRL_START_GDR** **/**< Current frame is a choosen to start the GDR activity. Applicable** *** when intraRefreshMethod is IH264_INTRAREFRESH_GDR** ***/**

	} IH264ENC_Control;

IH264ENC_CTRL_START_GDR: By making the control call with this value, the GDR activity gets enabled from that frame onwards. This will be effective only if intraCodingPreset = IH264_INTRACODING_USERDEFINED and intraRefreshMethod = IH264_INTRAREFRESH_GDR.

IH264ENC_SliceCodingParams: This structure controls the slice coding parameters. This structure is part of create and dynamic parameters structure. The changes between product 1.0 and product 2.0 w.r.t this structure are not major but only the data type of parameter sliceUnitSize has been changed from XDAS_Int16 to XDAS_Int32

Product 1.0	Product 2.0
XDAS_Int8 sliceCodingPreset ; XDAS_Int16 sliceMode ; **XDAS_Int16 sliceUnitSize ;** XDAS_Int8 sliceStartOffset [IH264ENC_MAX_NUM_SLICE_START_OFFSET] ; XDAS_Int8 streamFormat ;	XDAS_Int8 sliceCodingPreset ; XDAS_Int16 sliceMode ; **XDAS_Int32 sliceUnitSize ;** XDAS_Int8 sliceStartOffset [IH264ENC_MAX_NUM_SLICE_START_OFFSET] ; XDAS_Int8 streamFormat ;

5.5 Compatibility values for the newly added parameters in product 2.0

By setting the values of new parameters as mentioned in the below table, one can nearly generate the stream matching to that of product 1.0 generated. The values of the common parameters in product 1.0 and product 2.0 remain intact.

Parameter	Value
Create Parameters: IH264ENC_SVCCodingParams svcCodingParams; typedef struct IH264ENC_SVCCodingParams { XDAS_UInt8 **svcExtensionFlag**; XDAS_UInt8 dependencyID; XDAS_UInt8 qualityID; XDAS_UInt8 enhancementProfileID; XDAS_UInt8 layerIndex; XDAS_Int8 refLayerDQId; } IH264ENC_SVCCodingParams; XDAS_Int8 **enableWatermark** ; XDAS_Int32 **LTRPPeriod** ;	 **0** (disable) If svcExtensionFlag is disabled, rest all parameters in this structure are don't care. **0** (disable) **0** (disable)
InArgs: XDAS_Int32 **processId ;** IH264ENC_RoiInput **roiInputParams;** XDAS_UInt32 **inputKey ;**	 **Don't care** - Used only in multi-channel case **Don't care** - These parameters are read only if dynamicParams->enableRoi !=0 **Don't care** - This value is read only if the CreateParams->enableWatermark !=0

Parameter	Value
Dynamic Parameters: **IH264ENC_IntraCodingParams ;** typedef struct IH264ENC_IntraCodingParams { XDAS_Int8 intraCodingPreset ; XDAS_Int16 lumaIntra4x4Enable ; XDAS_Int16 lumaIntra8x8Enable ; XDAS_Int8 lumaIntra16x16Enable ; XDAS_Int8 chromaIntra8x8Enable ; XDAS_Int8 chromaComponentEnable ; XDAS_Int8 intraRefreshMethod ; XDAS_Int16 intraRefreshRate ; XDAS_Int16 **gdrOverlapRowsBtwFrames** ; XDAS_Int16 constrainedIntraPredEnable ; XDAS_Int8 **intraCodingBias** ; } IH264ENC_IntraCodingParams ; XDAS_Int32 **enableROI**;	 **0** **0**,IH264ENC_INTRACODINGBIAS_DEFAULT **0**(Disable ROI)
Miscellaneous Structures: **IH264ENC_InterCodingParams;** XDAS_Int8 **meAlgoMode** ; **IH264ENC_RateControlParams ;** XDAS_Int16 **minPicSizeRatioI** ; XDAS_Int16 **maxPicSizeRatioI** ; XDAS_Int16 **minPicSizeRatioP** ; XDAS_Int16 **maxPicSizeRatioP** ; XDAS_Int16 **minPicSizeRatioB** ; XDAS_Int16 **maxPicSizeRatioB** ; XDAS_Int32 **VBRDuration** ; XDAS_Int8 **VBRsensitivity** ; XDAS_Int16 **skipDistributionWindowLength**; XDAS_Int16 **numSkipInDistributionWindow**; XDAS_Int8 **enableHRDComplianceMode** ; XDAS_Int32 **frameSkipThMulQ5** ; XDAS_Int32 **vbvUseLevelThQ5** ;	 **0**,IH264ENC_MOTIONESTMODE_DEFAULT **0** **640** **0** **0** **0** **0** **8** **0** **5** **1** **1** **0** **0**

5.6 Extra Memory Requirements for Product 2.0

As product 2.0 has few additional/enhanced features, the memory requirement has increased compared to product 1.0 line. For more details, please refer document H264 Encoder Data Sheet - Table: Memory Statistics of Media Controller.