TriMedia Motion JPEG Decoder (VdecMjpeg) API

Note

This component library is not included with the TriMedia SDE but is available under a separate licensing agreement. Please contact your TriMedia sales representative for more information. ◆

Motion JPEG Decoder API Overview

Motion JPEG (MJPEG) is an implementation of JPEG for a sequence of video frames. As of now no known industry-wide standard exists. The MJPEG Decoder Library (VdecMjpeg) available with this release implements standard JFIF (JPEG File Interchange Format File), Motion JPEG format A (MJPEG-A) and Motion JPEG format B (MJPEG-B) decoding for baseline sequentially encoded frames. The current implementation of MJPEG decoder supports bitstreams with:

- LossyJPEG compression (DCT plus Huffman run length encoder).
- Bit stream from input images with 8-bit precision.
- Image formats: Monochrome and YCbCr formats (4:2:2 and 4:1:1).

See Pennebaker & Mitchell, “JPEG Still Image Data Compression Standard”, Van Nostrand-Reinhold NY, 1993 for more details on JPEG and “Motion JPEG Format”, Draft 2, April 15 1996, courtesy Apple Computer Inc. for details on MJPEG Formats A and B.

MJPEG-A is in full compliance with the ISO JPEG specification. Each frame contains two fields, with the first one being the odd field. Each field is a standard JPEG stream. More than one frame in a file makes a MJPEG Sequence file. In addition to standard JFIF markers (JPEG file interchange format), MJPEG-A adds a new application marker called APP1 (id = “ff e1”). MJPEG-B is nothing but Motion JPEG A stripped of all markers. The various fields of the APP1 marker are given below:

1. Unused: typically 0000
2. Tag: It should contain “mjpeg”
3. Field size: size of image data
4. Padded field size
5. Offset to next field
6. DCT Quantization table offset
7. Huffman Table Offset
8. Start of Image Offset
9. Start of Scan Offset
10. Start of Data Offset

All fields are 4 bytes and in Big Endian order.

This library provides a standard set of seven APIs, like other TriMedia components that conform to the TriMedia Software Streaming Architecture (TSSA). All interfaces and data structures are fully compliant with this architecture. The component takes a stream of tmAvPackets as input and produces a stream of tmAvPackets (YUV data) at the output.

The Application Library component (tmalVdecMjpeg) provides the basic functionality of OS independent JPEG decoding, while the operating system application library (OL) component (tmolVdecMjpeg) takes care of all inputs and outputs.

Performance

The typical performance obtained is around 8Mbits/sec of encoded bit stream, as measured on a cycle accurate simulator with known software tuning on a 100 MHz tml processor

Demonstration Programs

The VdecMjpeg component is normally used through the OL layer (tmolVdecMjpeg). An example program exolVdecMjpeg.c is provided to illustrate the use of this component. It takes an MJPEG file as an input stream, decodes it and puts it to VO. It uses the File Reader component to open and stream the data as packets to the VdecMjpeg component. After processing, the VdecMjpeg component streams packets of Yuv data to the VrendVO component which would in turn put it onto the VideoOut.

Overview of the tmolVdecMjpeg / tmalVdecMjpeg Component

The tmolVdecMjpeg component layer takes care of properly passing OS dependent parameters like empty and full Queue Id's to the tmalVdecMjpeg component. Default API's are provided and the API implementation is also largely the same as the default implementations provided with tsaDefaults.c.

The tmalVdecMjpeg component library provides an interface consistent with TSSA and provides in all, six C callable functions. A typical Usage Sequence will be:

1. tmalVdecMjpegGetCapabilities to get the decoder capabilities data structure.
2. tmalVdecMjpegOpen. This opens an instance of the decoder. The decoder does not put any restriction on the number of instances.
3. tmalVdecMjpegInstanceSetup. This registers the setup parameters provided by the user into internal instance variables.
4. tmalVdecMjpegStart. This decodes MJPEG frames sequentially until tmalVdecMjpegStop is called.
5. tmalVdecMjpegStop. This changes the `state` variable to STOP.
6. tmalVdecMjpegClose. This invalidates the instance and frees all memory created by the component.

Input Description

The MJPEG Decoder always operates in data streaming mode. It requests packets of data (default size 4k) using the `datain` callback function registered at the time of setup. Input packets are of the type `tmAvPacket_t`. Packet requests are made from within the `tmalVdecMjpegStart`.
The first packet received by the component should be aligned to a MJPEG Chunk. Three types of MJPEG Chunks are recognized by the decoder

- JFIF
- Motion JPEG A
- Motion JPEG B

The first two bytes of JFIF and MJPEG-A formats are “ff” and “d8”.

Output Description

Output packets are of the type `tmAvPacket_t`.

The sizes of the image that is being decoded are embedded within the input stream. In order to create the necessary buffers for the AvPackets and in order to set up the renderer, image sizes and format are to be communicated back to the user. To do this, the user creates a variable of type `ptmalVdecMjpegImageDescription_t` and registers it through the setup variable. The VdecMjpeg component will update this variable immediately after decoding the image description. The first `datain` call for an empty output packet will occur after this. It is the responsibility of the user to have created the buffers before passing the empty packets to the component. Typically the user will poll the `Initialized` field of `ImageDescription` to find out whether the MJPEG Decoder has decoded the image sizes. The user then creates the buffers and puts them into the empty queue. The VdecMjpeg uses these buffers to fill decoded data and puts them out through the `dataout` function.

Stopping the VdecMjpeg Component

The `VdecMjpeg` component may be stopped by either changing the `MjpegStates` variable to `MJPEG_STOP` from the AL layer or by calling tmolVdecMjpegStop. The former will stop the component after the current frame is processed. The latter is implemented by a call to the tsaDefaultStop. The example exolVdecMjpeg.c illustrates one way of stopping the processing chain. When the VdecMjpeg stops, it calls its completion function which may be used to synchronize with the other components.

Motion JPEG Decoder API Data Structure Descriptions

This section describes all the data structures concerned with the VdecMjpeg component

tmalVdecMjpegStates_t

```
typedef enum {
       MJPEG_RUN,
       MJPEG_STOP,
       MJPEG_PAUSE,
       MJPEG_SKIP
} tmalVdecMjpegStates_t,*ptmalVdecMjpegStates_t;
```

Fields

`MJPEG_RUN`	Value of the instance variable's `state` field when the decoder is decoding the stream. `tmalVdecMjpegStart` puts the component into this state.
`MJPEG_STOP`	Value of the `state` field when the decoder is stopped or is required to be stopped at the end of the current frame.
`MJPEG_PAUSE`	Reserved for future use.
`MJPEG_SKIP`	Value of the `state` field when the user wants to skip the current frame. It is the user's responsibility to release SKIP and put back RUN or STOP. This can be done by using the `progress` function.

tmalVdecMjpegStream_Type

```
typedef enum {
       MJPEG_A,
       MJPEG_B,
       MJPEG_JFIF,
       MJPEG_UNSUPPORTED_STREAM_TYPE
} tmalVdecMjpegStream_Type;
```

Fields

`MJPEG_A`	Encoded stream type is Motion JPEG-A.
`MJPEG_B`	Encoded stream type is Motion JPEG-B.
`MJPEG_JFIF`	Encoded stream type is JFIF.
`MJPEG_UNSUPPORTED_STREAM_TYPE`	Unknown input stream.

Description

These are the stream types used by the decoder internally. This is passed to the user through the (`ptmalVdecMjpegImageDescription_t`) ImageDescription field, of the setup variable. Necessary control action can be initiated by the user.

tmalVdecMjpegCapabilities_t,tmolVdecMjpegCapabilities_t

```
typedef struct{
       ptsaDefaultCapabilities_t     defaultCapabilities;
} tmalVdecMjpegCapabilities_t, *ptmalVdecMjpegCapabilities_t;
```

Fields

`defaultCapabilities`	Pointer to `tsaDefaultCapabilities_t`.

Description

See tsa.h for details. Replace al by ol in information above to get description of tmolVdecMjpegCapabilities.t.

tmalVdecMjpegImageDescription_t

```
typedef struct{
      Int32                         ImageHeight;
      Int32                         ImageWidth;
      Int32                         ImageStride;
      Int32                         PaddedImageHeight;
      tmVideoRGBYUVFormat_t         ImageFormat;
      Bool                          Initialized;
} tmalVdecMjpegImageDescription_t, *ptmalVdecMjpegImageDescription_t;
```

Fields

ImageHeight	Actual image height.
ImageWidth	Actual image width.
ImageStride	Calculated width of the Image Buffer based on the required granularity of Output Stride. See `tmalVdecMjpegInstanceSetup_t`.
PaddedImageHeight	Image height for which the bit stream is encoded. This can be larger than `ImageHeight` to take care of image heights which are not a multiple of eight.
ImageFormat	Decoded image format, one of vdfMono, vdfYUV420Planar, or vdfYUV422Planar.
Initialized	Set to True after the decoder fills in the other fields.

Description

This is the image description extracted from the encoded stream.

Note

The user is expected to create the image buffers and pass it to the decoder. The expected size of the buffer is the product of `PaddedImageHeight` times `ImageStride`. The user can poll the Initialized field to know when to create these buffers. ◆

tmalVdecMjpegInstanceSetup_t, tmolVdecMjpegInstanceSetup_t

```
typedef struct{
      ptsaDefaultInstanceSetup_t   default_setup;
      Bool                         littleEndian;
      Int32                        granularityOfOutputAddress;
      Int32                        granularityOfOutputStride;
      ptmalVdecMjpegStates_t       state;
      ptmalVdecMjpegImageDescription_tImageDescription;
} tmalVdecMjpegInstanceSetup_t, *ptmalVdecMjpegInstanceSetup_t;
```

Fields

default_setup	Pointer to a `tsaDefaultInstanceSetup_t` variable. See tsa.h.
littleEndian	True if the component is required to work in little endian mode, otherwise it is False. Currently only the compile time selection is employed.
granularityOfOutputAddress	The address alignment required for the output decoded image buffers (Y). Typically 64 for `Vrend` and 128 for `Vtrans`.
granularityOfOutputStride	Image width alignment due to output hardware constraints. Typically Nil for `Vrend` and 64 for `Vtrans`.

Description

This is the `InstanceSetup` struct for the MJPEG decoder. Replace al by ol in information above to get description of tmolVdecMjpegInstanceSetup.t.

tmalVdecMjpegProgressFlags_t

```
typedef enum{
       MJPEG_REPORT_FORMAT,
       MJPEG_REPORT_FIELD,
       MJPEG_REPORT_FRAME
       MJPEG_REPORT_STOP,
       MJPEG_REPORT_EOF
} tmalVdecMjpegProgressFlags_t, *ptmalVdecMjpegProgressFlags_t;
```

Fields

`MJPEG_REPORT_FORMAT`	Causes the `progress` function to be called immediately after the decoder extracts `ImageDescription` from the stream.
`MJPEG_REPORT_FIELD`	Causes the `progress` function to be called after each field has been decoded.
`MJPEG_REPORT_FRAME`	Causes the `progress` function to be called after each frame has been decoded.
`MJPEG_REPORT_STOP`	Causes the `progress` function to be called while stopping.
`MJPEG_REPORT_EOF`	Causes the `progress` function to be called when dataSize of the input packrt is lesser than the bufSize. This can be treated as a warning for an end of input stream.

Motion JPEG Decoder API Function Descriptions

This section describes the various API functions for the VdecMjpeg component.

tmalVdecMjpegOpen, tmolVdecMjpegOpen

```
tmLibappErr_t tmalVdecMjpegOpen(
      Int                          *instance
);
```

Parameters

instance	Pointer to the instance.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_MEMALLOC_FAILED	Returned if unable to allocate memory for decompressor.

Description

This function will create an instance of the VdecMjpeg component.

tmalVdecMjpegStart, tmolVdecMjpegStart

```
tmLibappErr_t tmalVdecMjpegStart(
      Int                          instance
);
```

Parameters

instance	Instance value assigned at the call of the MJPEGOpen function.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_DATAIN_FAILED	Returned if datain callback function fails.
TMLIBAPP_ERR_DATAOUT_FAILED	Returned if dataout callback function fails.
MJ_ERR_INVALID_PACKET	Returned if the input or output packets received through the empty / full queues are of improper format or size.
MJ_ERR_CORRUPT_STREAM	Returned if the Huffman encoded bitstream yields invalid states or symbols.

Description

This function will sequentially decode all frames from a MJPEG file.

tmalVdecMjpegStop, tmolVdecMjpegStop

```
tmLibappErr_t tmalVdecMjpegStop(
      Int                          instance
);
```

Parameters

instance	Instance value assigned at the call of the VdecMjpeg Open function.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the instance has not been opened.
TMLIBAPP_ERR_NOT_SETUP	Returned if the instance has not been setup.

Description

tmalVdecMjpegStop merely changes the components state variable to STOP.

tmolVdecMjpegStop calls tsaDefaultStop.

tmalVdecMjpegClose, tmolVdecMjpegClose

```
tmLibappErr_t tmalVdecMjpegClose(
      Int                          instance
);
```

Parameters

instance	Instance value assigned at the call of the MJPEGOpen function.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the passed parameter is not a valid instance.

Description

This function invalidates the instance and frees all memory allocated for instance variables and Decompression Instance.

tmalVdecMjpegGetCapabilities, tmolVdecMjpegGetCapabilities

```
tmLibappErr_t tmalVdecMjpegGetCapabilities(
      ptmalVdecMjpegCapabilities_t *capabilities
);
```

Parameters

capabilities	Pointer to a `tmalVdecMjpegCapabilities_t` data struct.

Return Codes

`TMLIBAPP_OK`	Returned on successful completion.
`MJ_ERR_NULL_POINTER`	Returned if `capabilities` is equal to Null.

Description

This function initializes the `capabilities` struct with the MJPEG Decoder component's values.

tmalVdecMjpegInstanceSetup, tmolVdecMjpegInstanceSetup

```
tmLibappErr_t tmalVdecMjpegInstanceSetup(
      Int                          instance,
      ptmalVdecMjpegInstanceSetup_t setup
);
```

Parameters

instance	Instance value assigned at the call of the MJPEGOpen function.
setup	Pointer to the setup data struct.

Return Codes

TMLIBAPP_OK	Returned on successful completion.
TMLIBAPP_ERR_INVALID_INSTANCE	Returned if the instance value is not valid.
TMLIBAPP_ERR_INVALID_SETUP	Returned if any of the fields in the setup data struct are invalid.

Description

This function registers the setup parameters provided by the user into the internal instance variables.

tmolVdecMjpegGetInstanceSetup

```
tmLibappErr_t tmolVdecMjpegGetInstanceSetup(
      Int                          instance,
      ptmolVdecMjpegInstanceSetup_t setup
);
```

Parameters

instance	Instance value assigned at the call of the MJPEG Open function.
setup	Pointer to the Instance setup structure of VdecMjpeg.

Return Codes

TMLIBAPP_OK	Returned on successful completion

Description

This function assigns the pointer to `tmolVdecMjpegInstanceSetup_t` structure allocated by the MJPEG Open function to `setup`.