



**Motion**

**JPEG**

**Streaming**

**TransChip Confidential**

This document contains proprietary information and except with the written permission of TransChip Inc., such information shall not be published or disclosed to others or used for any purpose. The document shall not be duplicated in whole or in part.

**Update List:**

Rev	Change	Description	Reason for change	Done By	Date
0.1	Creation			Lior	25 Mar. 2004
0.2	Update API	Bit rate control	New feature	Lior	16 June 2004

## Table of Contents:

<b>1</b>	<b>Purpose .....</b>	<b>4</b>
<b>2</b>	<b>Definitions/ Terms .....</b>	<b>4</b>
<b>3</b>	<b>Applicable Documents .....</b>	<b>4</b>
<b>4</b>	<b>MJPEG description and requirements .....</b>	<b>5</b>
<b>5</b>	<b>Currently supported features .....</b>	<b>6</b>
<b>6</b>	<b>Cyclic queue .....</b>	<b>7</b>
6.1	Setting up the queue parameters: .....	8
<b>7</b>	<b>MJPEG API functions .....</b>	<b>10</b>
7.1	API functions .....	10
7.2	Structures:.....	11
<b>8</b>	<b>Application example.....</b>	<b>14</b>
<b>9</b>	<b>References .....</b>	<b>15</b>

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	3 of 15

## 1 Purpose

This document describes the implementation of MJPEG streaming to memory as well as application example and description of system limitations.

It can be used as a reference guide for implementing a real MJPEG streaming system.

## 2 Definitions/ Terms

MJPEG – Motion JPEG.

DMJPEG – Decompression of MJPEG file.

QF – Quality factor.

AVI<sup>1</sup> – Audio Video file format.

BB – Base Band (also referred to as “the system”).

DSC – Digital Still Camera.

OSD – On Screen Display.

FW – Firmware.

HW – Hardware.

SW – Software.

BPP – Bit Per Pixel.

TC – TransChip.

QT – Quantization Tables.

PVI – Parallel Video Interface.

## 3 Applicable Documents

TC5747 Programmer's reference.

---

<sup>1</sup> AVIs are Microsoft's approach to storing movies. It uses the [RIFX](#) format to store information in a tree structure. Both audio and video data are encoded using the Windows' installable codecs. Please note that TransChip doesn't support audio capture.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	4 of 15

#### **4 MJPEG description and requirements**

Motion JPEG video file is essentially a collection of JPEG files bundled together and wrapped up in an AVI file format.

The following list describes the requirements of the M-JPEG capture system:

- Standard M-JPEG AVI file format so the captured video can be played on PC.
- Queue of frames to solve real-time issues and allow BB momentarily CPU busy peaks.
- 20 fps for VGA, 30 fps for QVGA and below.
- Timestamp OSD.
- Thumbnail of the first frame.
- Up to 30 fps decompression of M-JPEG.
- Single JPEG capture during M-JPEG.
- Zoom change during M-JPEG capture.
- Image controls changed during M-JPEG capture.
- Fixed frame rate.
- Fixed bit rate (bit rate control).
- Downscale M-JPEG movie during decompression to PVI.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	5 of 15

## **5           Currently supported features**

- Standard M-JPEG AVI file format so the captured video can be played on PC.
- Queue of frames to solve real-time issues and allow BB momentarily CPU busy peeks.
- 10 fps for VGA, 15 fps for QVGA and below.
- Timestamp OSD.
- Up to 30 fps decompression of M-JPEG.
- Single JPEG capture during M-JPEG (Application implementation)
- Image controls changed during M-JPEG capture.
- Fixed frame rate.
- Fixed quantization tables (fixed quality) or bit rate control algorithm.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	6 of 15

## 6 Cyclic queue

In order to support a real-time system and allow the host busy peeks without losing frames, a cyclic queue was implemented with the interface that allows the host to change the queue parameters.

The queue parameters that can be changed are:

1. The buffer size of 1 element in the queue.
2. The number of buffers in the queue.

Those parameters need to be set according to the system limitations and the host requirements of the application. In the following example (case study) we will illustrate such limitations, application definition and the conclusion regarding the queue settings.

Consider an application where M-JPEG video clip of 128x120 is to be captured in 10 fps.

The system limitations are (BB limitations):

- Max file size: 64KB.
- Interface bandwidth: 25KB per second (in average<sup>2</sup>).

Calculations:

In order to support the requested 10 fps on the given system (25KB/sec) we need to limit each frame to a maximum of 2.5KB.

If all the frames will be 2.5KB and the maximum file size (limited by the system) is 64KB then the max duration of the video clip will be ~2.5 seconds (64/25).

We can give another application requirement that will limit the frame size:

Say the application is to capture 10 seconds of video in 10fps. This will lead into 100 frames to capture. With the previous system limitation we have placed (max 64KB per file) we can conclude the each frame needs to be around 655 bytes (64KB / 100).

Since MJPEG header of each frame is 200 bytes we can now calculate the bit rate of the above calculated frame sizes:

---

<sup>2</sup> The CPU load and the theoretical interface bandwidth (with or without DMA) need to be considered together to derive the average interface bandwidth of the system. E.g. a system with 50KB per second interface and CPU load of 50% will only be able to read 25KB per second (unless DMA is used).

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	7 of 15

2.5KB per frame:

$2.5 \times 1024 - 200 = 2360$  bytes of JPEG code representing 128x120 image.

$128 \times 120 = 15360$  pixels.

Hence the above shows a  $2360 \times 8 \text{bit} / 15360 = 1.23 \text{ bpp}$ . (high quality JPEG).

655 bytes per frame:

$655 - 200 = 455$  bytes, Hence  $455 \times 8 \text{bit} / 15360 \approx 0.24 \text{ bpp}$  (Very low quality JPEG).

As you can see some tradeoffs needs to take place and we need to decide and prioritize the supported features. We can lower the frame rate or capture smaller images in order to increase the quality or allow longer video clips durations.

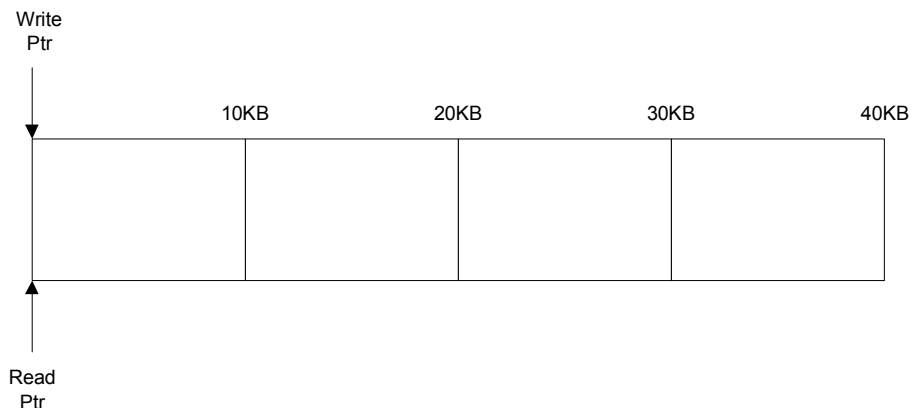
## 6.1 Setting up the queue parameters:

Say we have agreed on an average of 1.5-2K per image, we now need to set the queue buffer size.

As the current implementation doesn't support fixed frame size (i.e. bit rate control) we need to allow buffer size that is 5 times the average frame in order to allow rare high frequency images to be placed on the buffer without overlapping the next frame area. If such case would happen the next captured frame will overlap the previous end of frame and the result can be corrupted video file.

Hence, we will set the buffer size to 10K and initiate number of buffers as 4. This setup assumes that the maximum JPEG file (still JPEG file) with OSD initialized is 40KB. The reason we use maximum still JPEG file size is because both still JPEG and M-JPEG use the same memory and share the same limitations.

Initially a queue that was setup as 10KB per buffer with 4 buffers will look like this:

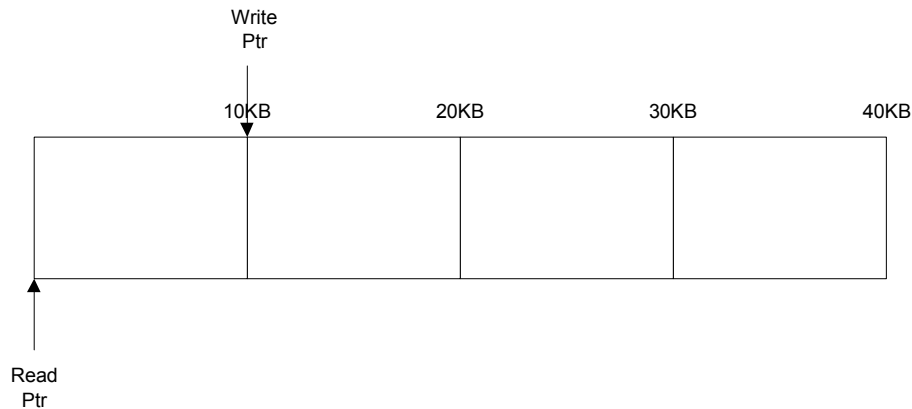


Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	8 of 15



The write and read pointers are initialized to the start address of the queue. When the DSC starts capture the video clip the frames are written into the write address and after each frame the write pointer is incremented by a buffer size and prepared for the next frame to be captured.

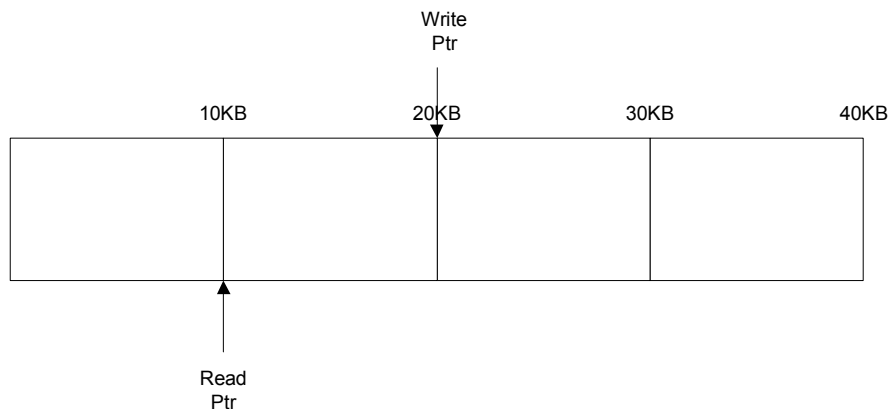
After 1 frame the queue pointers will be located as bellow:



During the capture phase the host (BB) polls the write pointer and detects that the write pointer was incremented and knows its time to read the first frame of the M-JPEG clip.

If the host is fast enough it will read the frame and increment the read pointer by the buffer size before the DSC captured the next frame. On a slower system it will read the frame, increment the read pointer and find that the write pointer has incremented again and there is another frame ready in the queue.

In this case, after reading the first frame the queue will look like this:



Please note that “faster” and “slower” systems that where referenced above are relative to the desired frame rate. A “slower” system in 15 fps can be quite fast if the desired frame rate was 5 fps. Every system needs to be evaluated according to the application requirements.

As the name implies (cyclic buffer) when the write pointer reaches the end is wraps around to the start and writes again over the first buffer.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	9 of 15

If the DSC reaches a state where the write pointer has looped around and reached the read pointer it stops capture and raises a flag full. When the BB reads the flag full and finds that its "true" it indicates that at least 1 frame was dropped (there is no indication how many frames were dropped since this situation will not occur on a real system).

When such condition is reached, the system needs to be tuned and re-defined so it will never occur on a real-life situation.

In order to avoid such case (when it happens) we can lower the frame rate, lower the quality factor or even consider to capture smaller frames (in terms of width and height). Furthermore, the queue setup maybe tuned e.g. allocate more buffers to compensate BB MIPS overload peaks.

## 7 MJPEG API functions

### 7.1 API functions

- `int TCmpegQuality(unsigned short usQuality);`  
Sets the MJPEG quality factor. Accepts values from 1-99.
- `int TCmpeg2mem(TCmpeg2memStr *pTCmpeg2memInfo, unsigned long *pulStartAddress);`  
Used to initiate or stop a MJPEG capture to memory (streaming). It also returns the start address of the queue.

When initiating a MJPEG capture (bEnable = TRUE) you need to pass a description of the queue: Number of buffers in queue and the buffer size (of 1 element). E.g. if you know there are 50KB free to use in JPEG memory (see comment) and each frame in the MJPEG is targeted to be ~6K you can initiate the queue to 10K buffer size and 5 buffers in queue. For bit rate control initiate the target frame size and the max frame size according to the requested compression ratio.

Comment: The free JPEG memory to use is the same as the maximum JPEG capture limitations or the JPEG decompress limitations. Note that it depends on the OSD usage since the OSD data base is allocated on the JPEG memory block.

- `int TCmpegGetWriteCnt(unsigned short *pusWriteCnt);`  
Used to test the queue condition and decide if there is a new frame ready. (If write cnt is bigger than read cnt)
- `int TCmpegGetFlagFull(unsigned short *pusFlagFull);`  
Used to test the queue condition and decide if there is a queue full condition. If so it means we lost at least 1 frame and the system setting is not tuned correctly (we have real-time issues). If this condition happens frequently we need to decide either to capture with lower quality factor (use TCmpegQuality) or to lower the frame rate or capture smaller images (e.g. instead of QQVGA capture 128x120). All those parameters needs to be tuned during system integration.
- `int TCmpegSetReadCnt(unsigned short usReadCnt);`  
Used to set the queue read count.
- `int TCmpegClearFlagFull(void);`  
Clears the flag full state in the DSC

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	10 of 15

- `int TCdmjpegRequest(TCdmjpegInfoStr *pdmjpegInfo, unsigned short *pusMaxFileSize);`  
Used to initiate a MJPEG decompression process
- `int TCdjpegLoading(unsigned short* pusBuffer, unsigned short usWordCount, TCstate eState);`  
Allows JPEG file to be loaded into CORE memory for decompression. It may be used iteratively several times to load small parts of the file. In order to load a JPEG file the function has to be called at least twice. Once with eState=START\_TRANSFER and the second time with eState=END\_TRANSFER (so the CORE will know that the host has finished the memory accesses). If the file is to be loaded in chunks than the first call would be with START\_TRANSFER, the next few calls with CONT\_TRANSFER and the last call with END\_TRANSFER (with or without data).
- `int TCdmjpegDecompress(void);`  
Decompress the frame that was previously loaded by TCdjpegLoading.
- `int TCdmjpegEndProcess(void);`  
Ends a MJPEG decompression process.

## 7.2 Structures:

```
typedef struct
{
    unsigned short usOneBufferSize;
    unsigned short usNumOfBuffersInQueue;
    unsigned short bEnable;
    unsigned short bOSD;
    unsigned short usTargetFrameSizeInBytes;
    unsigned short usMaxFrameSizeInBytes;
    unsigned short usQualityFactor;
    unsigned short bUseFixedQualityFactor;
}TCmjpeg2memStr;
```

As explained in previous chapter, when designing the MJPEG application it is important to know in advance the maximum frame size (according to quality and width-height) and the system bandwidth so the queue can be initialized properly.

As the structure elements names implies:

- usOneBufferSize – One buffer size. E.g. 10\*1024 (10K).
- usNumOfBuffersInQueue – Number of buffers in queue. E.g. 4 or 5.
- bEnable – Start or stops the capture process. Note that each start initialize all pointers and do not continue from the last time it captured MJPEG clip!
- bOSD – If TRUE, ALL OSD regions would be captured with image, otherwise only OSD frame would be captured (assuming OSD frame is enabled.).
- usTargetFrameSizeInBytes – The desired **average** frame size.
- usMaxFrameSizeInBytes – The maximum allowed frame size. If a frame size exceeds this value it will be discarded by the FW and will not pass to the application.
- usQualityFactor – Fixed quality factor
- bUseFixedQualityFactor – Boolean flag. If TRUE – use fixed QF, else bit rate control.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	11 of 15

```
typedef struct
{
    unsigned short usWidth;
    unsigned short usHeight;
    unsigned short usLumQTOffset;
    unsigned short usChromQTOffset;
    unsigned short usMJheaderSize;
} TCdmjpegInfoStr;
```

When decompressing a MJPEG clip (playback) we need to initiate the DSC and prepare it for decompression. The information it needs to be prepared for decompression is:

- usWidth – The frame width
- usHeight – The frame height
- usLumQTOffset – The offset from start of frame where the luminance QT are located. Currently fixed to 38 on TC DSC.
- usChromQTOffset – The offset from start of frame where the chrominance QT are located. Currently fixed to 103 on TC DSC.
- usMJheaderSize – The MJPEG header size. Currently fixed to 200 on TC DSC.

The QT offsets and the MJPEG header size are assumed to be fixed for all frames and thus this function needs to be called only once for each MJPEG clip playback. Moreover, if an external AVI file of MJPEG that was not produced by TC is to be used, the application needs to parse the header in order to extract the correct offsets.

### 7.3 Target and maximum frame size (bit rate control):

When fixed bit rate is required the Boolean `bUseFixedQualityFactor` in `TCmjpeg2memStr` structure must be FALSE. The frame size calculation depends on the system limitations and on the desired quality of the image.

A medium quality image may be resulted (most of the time) by a compression ratio of 1:16.

If we take for example 128x120 pixels we get:

128x120x2 = 30720 bytes (raw data YUV422 or RGB565).

Compressed: 30720/16 = 1920 bytes.

Hence we can conclude that for standard quality, target frame size should be 1920.

This number can be represented also as 1bpp (1 bit per pixel) because:

128x120 = 15360 pixels.

1920\*8 = 15360 bits.

15360/15360 = 1 bpp.

However, the net jpeg code size is not the real frame size. Each frame is accommodated by a 200 bytes header and hence the value that needs to be passed to the FW is 2120 (1920+200).

**Important notice: When bit rate control is used we cannot provide constant image quality and the image quality may vary dependent on the image content (image frequency).**

When a high frequency image is presented to the DSC (many details) it is harder to compress and hence the frame might rise above the requested frame size. The bit rate control algorithm will detect this situation and react by lowering the QF so the image will be compressed to the target frame size. This will look to the user as “low quality” image because of the “blockiness” and JPEG artifacts.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	12 of 15

As a thumb rule the maximum frame size (`usMaxFrameSizeInBytes`) may be set to twice the frame size.

Continuing our previous example we would set the maximum frame size to 4000.

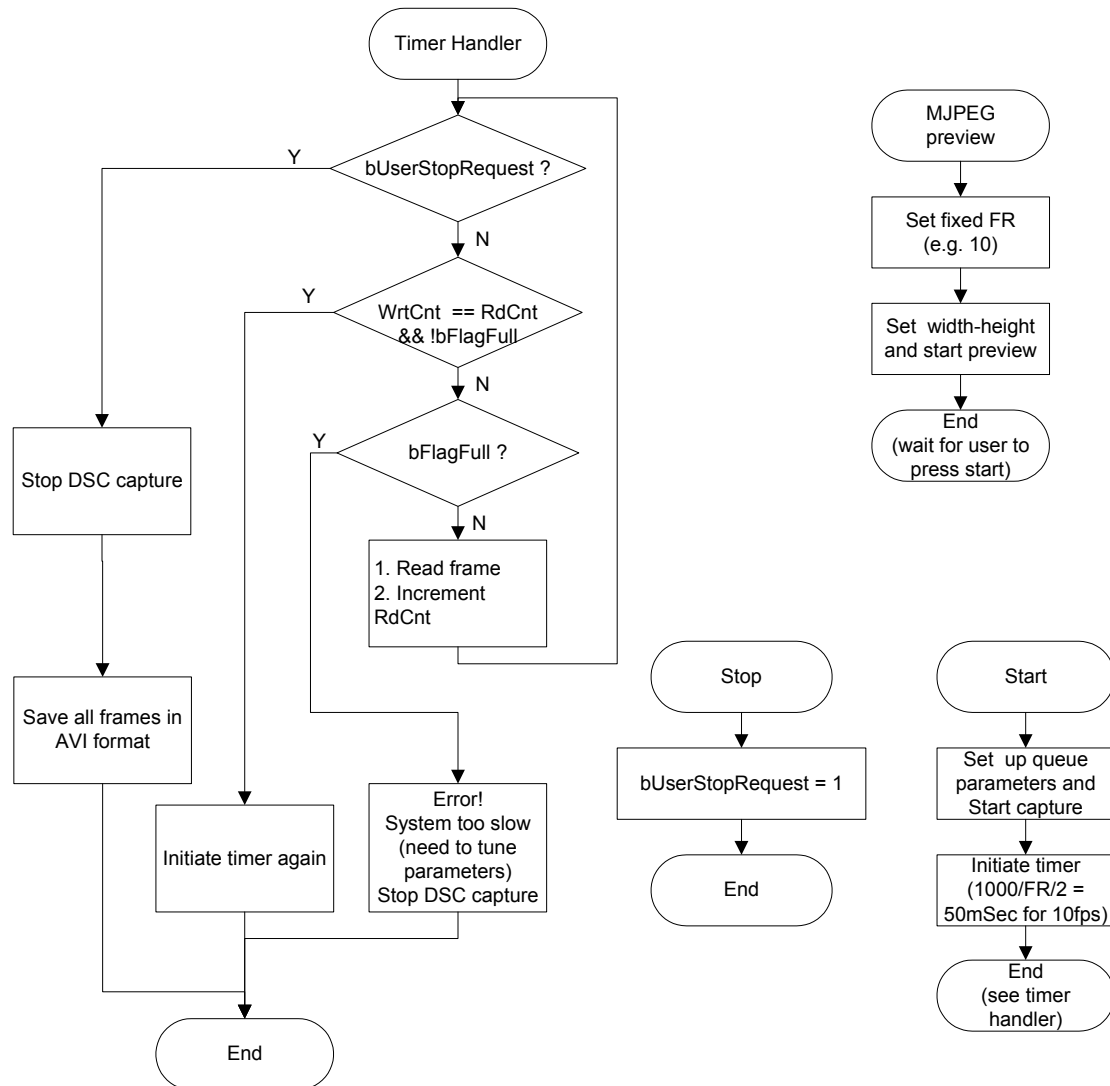
The reason for that is that we wouldn't want to loose frames frequently. When the image content changes rapidly there is no way to predict the future compression ratio of the next image and hence there might be peaks where the frame size grows by 50-80% of the target frame size. Still, the bit rate control algorithm will react immediately by lowering the QF and converge towards the desired frame size.

If we look at the average frame size it will be very close the requested frame size (+-3%).

Please note that the queue frame size should be set to be at least 4 times the target frame size. (In our example – at least 8KB.)

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	13 of 15

## 8 Application example



The MJPEG preview flowchart sets the preview conditions for MJPEG capture. It sets the fixed frame rate, preview width and height and starts the preview.

The “Start” flowchart (when user pressed start) sets the queue parameters and initiate a timer that will be twice as fast as the frame rate. For example the frame rate is set to 10 (cycle time is 100mSec) the timer will be set to at least 50mSec.

The “Stop” flowchart (when user press stop) just raises a flag to be used in the timer handler.

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	14 of 15

The Timer Handler flowchart looks on the user request flag and stops the capture process gracefully if the user pressed stop. If not it checks whether there are new frames in the queue. If the write counter is different than the read counter (can only be bigger) and the flag full is not true it means that there is a frame ready to be read out. When there is a frame ready it reads the frame, increments the read counter and goes back to the beginning to see if there is another frame ready. If there is no frame ready it initiates the timer again and the whole process will start again after the timer will expire again.

If a flag full condition is detected you can either decide to stop the whole process or to acknowledge that some frames were dropped (at least one) and to continue the process by reading the last frame and set the flag full to false – this will make the DSC to continue the capture (When a flag full condition is reached in the DSC, it stops capturing and it doesn't trash the frames in the queue buffers.)

## 9        **References**

- <http://www.daubnet.com/formats/RIFF.html>
- <http://www.daubnet.com/formats/AVI.html>

Classification:	Document Title:	Written By / Owner	Creation Date	Page
TransChip Ref Design	MJPEG reference design	Lior Weintraub	25 March 2004	15 of 15