

Still picture Grabbing with MB86291

© Fujitsu Microelectronics Europe GmbH

History

10-10-03	MMi	V1.0	First version

Warranty and Disclaimer

To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH restricts its warranties and its liability for **all products delivered free of charge** (eg. software include or header files, application examples, application Notes, target boards, evaluation boards, engineering samples of IC's etc.), its performance and any consequential damages, on the use of the Product in accordance with (i) the terms of the License Agreement and the Sale and Purchase Agreement under which agreements the Product has been delivered, (ii) the technical descriptions and (iii) all accompanying written materials. In addition, to the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH disclaims all warranties and liabilities for the performance of the Product and any consequential damages in cases of unauthorised decompiling and/or reverse engineering and/or disassembling. **Note, all these products are intended and must only be used in an evaluation laboratory environment.**

1. Fujitsu Mikroelektronik GmbH warrants that the Product will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of receipt by the customer. Concerning the hardware components of the Product, Fujitsu Mikroelektronik GmbH warrants that the Product will be free from defects in material and workmanship under use and service as specified in the accompanying written materials for a duration of 1 year from the date of receipt by the customer.
2. Should a Product turn out to be defect, Fujitsu Mikroelektronik GmbH's entire liability and the customer's exclusive remedy shall be, at Fujitsu Mikroelektronik GmbH's sole discretion, either return of the purchase price and the license fee, or replacement of the Product or parts thereof, if the Product is returned to Fujitsu Mikroelektronik GmbH in original packing and without further defects resulting from the customer's use or the transport. However, this warranty is excluded if the defect has resulted from an accident not attributable to Fujitsu Mikroelektronik GmbH, or abuse or misapplication attributable to the customer or any other third party not relating to Fujitsu Mikroelektronik GmbH.
3. To the maximum extent permitted by applicable law Fujitsu Mikroelektronik GmbH disclaims all other warranties, whether expressed or implied, in particular, but not limited to, warranties of merchantability and fitness for a particular purpose for which the Product is not designated.
4. To the maximum extent permitted by applicable law, Fujitsu Mikroelektronik GmbH's and its suppliers' liability is restricted to intention and gross negligence.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES

To the maximum extent permitted by applicable law, in no event shall Fujitsu Mikroelektronik GmbH and its suppliers be liable for any damages whatsoever (including but without limitation, consequential and/or indirect damages for personal injury, assets of substantial value, loss of profits, interruption of business operation, loss of information, or any other monetary or pecuniary loss) arising from the use of the Product.

Should one of the above stipulations be or become invalid and/or unenforceable, the remaining stipulations shall stay in full effect.

1. Introduction

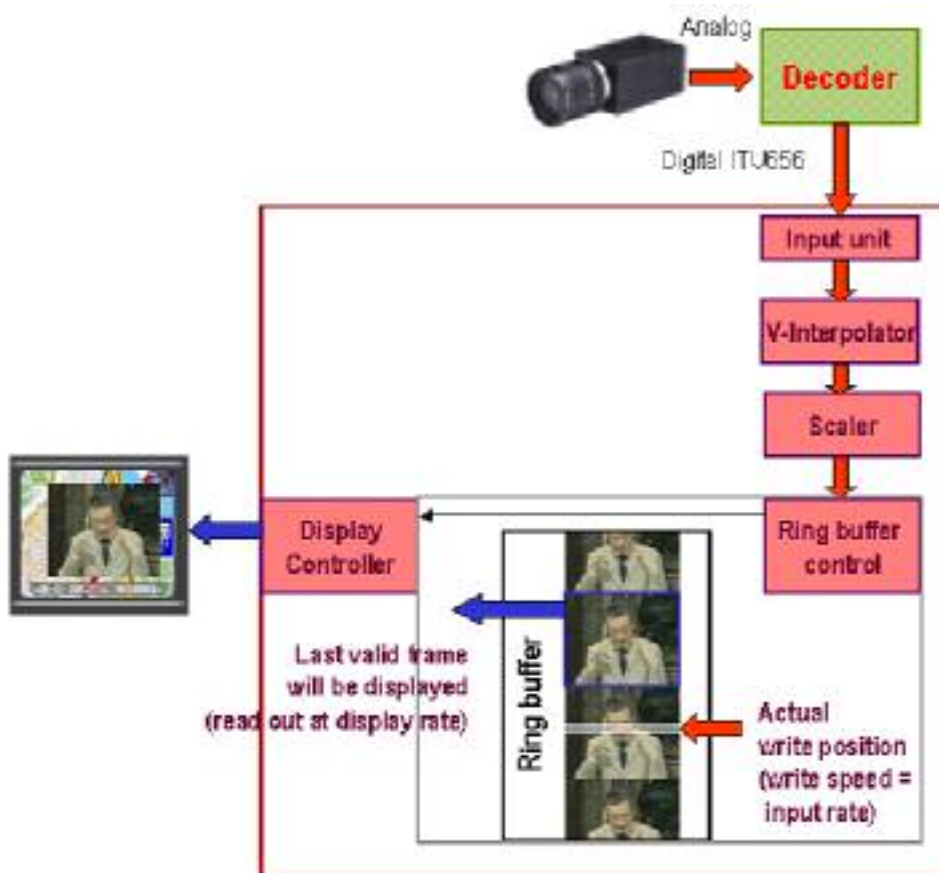
In many applications, pictures from the video input stream needs to be read back to the connected CPU for further processing ("video-grabber applications"). Since many Graphic Controllers of the Fujitsu Lineup has a powerful video input unit on-chip which reads in the video-stream to a memory mapped buffer, these devices can be used for this purpose. However, please note that the actual intention for these video-inputs was to simply read and display running video streams. Therefore, creating grabber-applications for the relevant devices from the Cremson family (MB86291 Scarlet and MB86292 Orchid), you need to take care about some special conditions which are described in this application note.

2. The video-input unit - Principle of operation

The video input unit of Scarlet and Orchid was designed to accept PAL or NTSC streams in the format ITU656 and to scale and display them as moving picture on the W(indow) layer. In addition, automatic de-interlacing and frame-rate conversion is available as features.

When the video input unit is initialized by the appropriate registers, the unit constantly reads in and stores frame by frame into the assigned ring buffer with the input frame rate (eg. PAL=50fps –odd and even frames).

On the other side, the display controller reads out the last complete frame from this buffer and displays it on the Window-layer. The display controller updates the window layer in accordance to the display rate (e.g. TFT=60Hz).



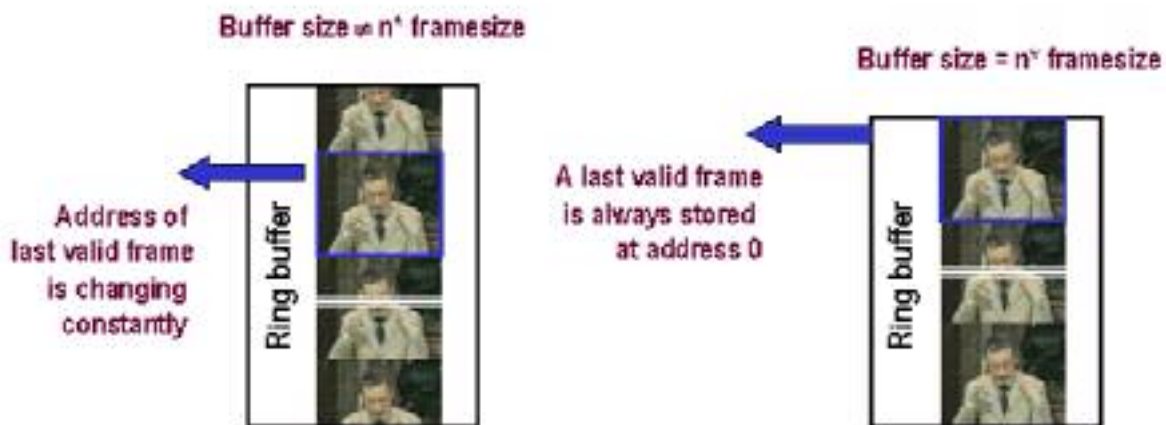
Scarlet video input processing

3. Grabbing single frames out of the Ring Buffer

As long as the video input works in the way described above, there is nothing to do for the user. The whole process works on it's own without any interactions required. But if a single frame should be read out, there some aspects to consider:

First there is the fact, that the ring buffer will be filled with 50 or 60 frames per second. If the buffer itself is not very big, there is no chance that the CPU can read out one written frame without the risk that this frame gets overwritten a couple of times by new video data coming in. Therefore, the video input needs to be stopped during reading out frames !

Then, we have the fundamental problem, that in this GDC series the information where the last written (valid) frame in the buffer is stores is not disclosed to the user in a register (note that in the Coral series this is possible). Therefore we have to apply a small trick which works if we are not too worried about a frame loss of 1-2 frames in time : If we setup the buffer-size as exactly n times the size one grabbed frame will take inside the buffer, we can assume that the position of one frame always starts at position 0 in the buffer. So all we need to do is reading from 0 up to the size of one frame.



Assigning the buffer to n*framesize allows to always read one frame

Example for reading 128x128 picture frame with a buffer of 3 frames :

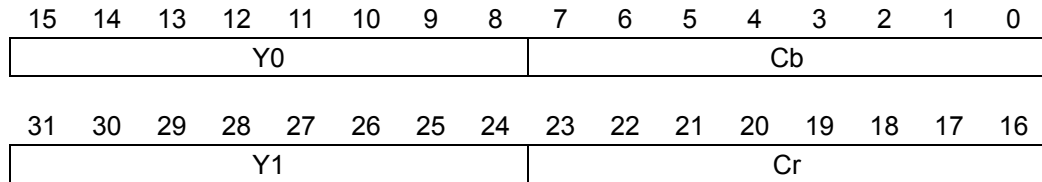
```
BUFSTART = 0x0A0000; // buffer start in Scarlet memory
BUFEND = (128*128*2)*3; // buffer end

GdcCapSetVideoCaptureBuffer(BUFSTART,BUFEND,0x14); // assign buffer
GdcCapSetImageArea(0,0,128,128); // set image area to 128x128
GdcCapSetVideoCaptureMode(0x83000002); // enable video input in PAL mode
setscaling(); // set the scaling values
```

Finally, there could be the problem that the video pictures are stored in YUV422 format (as they will be transmitted). Usually, pictures will be processed in RGB format on a CPU. Therefore, the YUV format needs to be converted to RGB.

The YUV422 format stores one chrominance (color) value for 2 consecutive pixels (but two luminance values).

If we read one 32-bit word from the video buffer, we get information for 2 pixels in the format :



The formula for converting these 2 pixels back to RGB is :

$$\begin{aligned}
 R0 &= Y0 && + 1.402 && * (Cr-128); \\
 G0 &= Y0 - 0.34414 && * (Cb-128) - 0.71414 && * (Cr-128); \\
 B0 &= Y0 + 1.772 && * (Cb-128) && ; \\
 R1 &= Y1 && + 1.402 && * (Cr-128); \\
 G1 &= Y1 - 0.34414 && * (Cb-128) - 0.71414 && * (Cr-128); \\
 B1 &= Y1 + 1.772 && * (Cb-128) && ;
 \end{aligned}$$

In order to avoid complex floating point calculations, it is also possible to normalize these equations to * 131072 (0x20000). The result is :

$$\begin{aligned}
 R0 &= (Y0*131072 && + 183763 && * (Cr-128))\gg 17; \\
 G0 &= (Y0*131072 - 45107 && * (Cb-128) - 93604 && * (Cr-128))\gg 17; \\
 B0 &= (Y0*131072 + 232260 && * (Cb-128) &&)\gg 17; \\
 R1 &= (Y1*131072 && + 183763 && * (Cr-128))\gg 17; \\
 G1 &= (Y1*131072 - 45107 && * (Cb-128) - 93604 && * (Cr-128))\gg 17; \\
 B1 &= (Y1*131072 + 232260 && * (Cb-128) &&)\gg 17;
 \end{aligned}$$

Leaving us with only integer arithmetics and shift operations. Finally we have to clamp the values to their min/max ranges :

```

if (R0 > 255) R0 = 255;
if (R1 > 255) R1 = 255;
if (G0 > 255) G0 = 255;
if (G1 > 255) G1 = 255;
if (B0 > 255) B0 = 255;
if (B1 > 255) B1 = 255;
if (R0 < 0) R0 = 0;
if (R1 < 0) R1 = 0;
if (G0 < 0) G0 = 0;
if (G1 < 0) G1 = 0;
if (B0 < 0) B0 = 0;
if (B1 < 0) B1 = 0;

```

As a last step, the results can be combined to a usual RGB-values. For example, if we would like to use the result of this operation to be used in the Scarlet's 16-bpp format, the final conversion would look like this :

```
pixel0= ((R0<<7) &0x7C00) + ((G0<<2) &0x3E0) + ((B0>>3) &0x1F) ;  
pixel1= ((R1<<7) &0x7C00) + ((G1<<2) &0x3E0) + ((B1>>3) &0x1F) ;
```

4. Further information

See the available application note “AN-MB86291-Videoscaler.pdf” for more details on the basic configuration and register settings of the Scarlet video input unit.

Also refer to the available source code examples for more details.

Note that the Coral P/PA series can handle the YUV-RGB conversion and other features such as information on the frame start addresses etc in hardware. This is basis for video texture mapping which is possible on these more advanced GDCs.