APPLICATION NOTE

Scan conversion using the SAA4998 (FALCONIC-EM)

Version 1

AN10233

Philips
Semiconductors

PHILIPS

Scan conversion using the SAA4998 (FALCONIC-EM)

Version 1

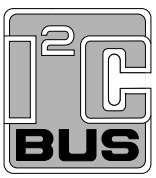
Application Note AN10233

Abstract

This application note describes the scan converter module MK14-EM. 50 or 60 Hz video input signals are converted to progressive scan or 100..120 Hz (scan rate doubling, $2f_H$ and $2f_V$). A software solution permits conversion from 50 Hz to 60 Hz or 75 Hz also, with the horizontal frequency being doubled ($2f_H$).

The module offers various analog inputs as well as a digital ITU-656 interface. Two color decoders permit dual-channel display as double-window or picture-in-picture. The Y-U-V output is analog again.

The ICs used on the board and described in this application note are the color decoder SAA7118, the scan converter SAA4979 and the motion compensation IC with embedded memories, the SAA4998. The memories in the SAA4998 either serve as field or frame memory for motion compensation, or one or both of them can be switched as buffer memory to synchronize the second input channel.



Purchase of Philips I^2C components conveys a license under the Philips I^2C patent to use the components in the I^2C system, provided the system conforms to the I^2C specifications defined by Philips.

APPLICATION NOTE

Scan conversion using the SAA4998 (FALCONIC-EM)

AN10233

Author:

Heinrich Waterholter
BL-MTS, S&A
Hamburg, Germany

Keywords

SAA4998, SAA4979, SAA7118,
motion compensation, motion estimation, natural motion
DNR (dynamic noise reduction),
EDDI, edge dependent deinterlacing,
peaking
DCTI

Date: April 30, 2003

Scan conversion using the SAA4998 (FALCONIC-EM)

Version 1

Application Note AN10233

Summary

The MK14-EM Improved Picture Quality (IPQ) module is a scan converter intended to convert TV scan rates of 50 / 60 Hz interlace to 50 / 60 Hz progressive, 75 / 90 Hz interlace or 100 / 120 Hz interlace. All modes are motion compensated. This feature eliminates motion judder which occurs whenever the source movement rate is different from the display rate.

Compared to previous Philips scan converter modules the MK14-EM employs the SAA4998 which is a motion compensation IC with embedded field memories. The memories serve as background field or frame memory for the motion compensation process, but can also be configured as buffer memories for the second simultaneous display channel. In this way the SAA4998 saves four external memories and therefore reduced chip count considerably.

An additional feature compared to previous motion compensation ICs is EDDI (edge dependent deinterlacing). This function analyzes the progressive output signal of the standard deinterlacer and in case of staircases along an edge replaces pixels to generate a smooth edge.

The application note describes the ICs and their picture improvement functions, gives details on circuit diagrams and layout of the board and supplies a register table for control by I^2C bus.

Table of Contents

Table of Figures

1. Introduction

The MK14-EM module is a scan converter for television input signals. The converter doubles the line frequency, the output field frequency depends on the mode. It can also be doubled, in this case 50 Hz PAL/SECAM or 60 Hz NTSC are changed to 100 Hz or 120 Hz. Or it can remain unchanged as is the case in progressive scan conversion (deinterlacing).

The module offers various analog input, but also can be configured to accept digital video data in ITU-656 format. The output signals are analog YUV. There are two color decoders on board enabling two video sources to be displayed at the same time, either in double window or PIP (picture-in-picture) mode.

The scan conversion IC on the board is the SAA4979. It decodes the digital video input streams, has a built-in scan conversion memory, offers various picture improvement functions and has D/A converters to generate the analog output signals. On-chip the SAA4979 also has a microcontroller with embedded RAM and ROM and an I²C interface for communication with a main controller.

Motion compensation on the converted signal is done by the SAA4998 (FALCONIC-EM). This IC has the core functions of the SAA4993 (FALCONIC), but also has two field memories on-chip (EM stands for 'embedded memories') which are needed for motion compensation. These memories can also be configured differently, in that case one or both of them serve as buffer memory for video data in the subchannel (PIP memory).

This application note describes the hardware functions of the SAA4979 and SAA4998 and the application environment needed to realize 100 Hz scan conversion as well as extra functions. A register command table is added (chapter 10), this is the control interface to an outside master controller or to a user who wants to define certain settings of the board.

2. Features of IPQ modules

Table 1 gives an overview of the MK14-EM scan converter modules. The individual modules are equipped with a SAA4998 or SAA4999 and one or two color decoders SAA7118.

Module 1: MK14-EM equipped with SAA4998 and 2 x SAA7118

Module 2: MK14-EM equipped with SAA4999 and 2 x SAA7118

Module 3: MK14-EM equipped with SAA4999 and SAA7118

module:	**1**	**2**	**3**
Feature	**MK14-EM with SAA4998**	**MK14-EM with SAA4999**	**MK14-EM with SAA4999 (no PIP)**
4:2:2 scan rate doubling (50 to 100 Hz or 60 to 120 Hz)	x	x	x
4:2:2 conversion to progressive scan (50 Hz / 60 Hz proscan)			
4:2:2 conversion 50 Hz to 75 Hz or 60 Hz to 90 Hz			
4:2:2 conversion 50 Hz interlace to 60 Hz progressive			
3.5 MBit embedded scan conversion memory	x	x	x
Embedded memories for motion compensation / PIP	2	1	
Sample rate conversion for linear zoom and compression	x	x	x
Panorama mode	x	x	x
Dynamic noise reduction	x	x	x
Noise estimator	x	x	x
Black bar detection	x	x	x
Luminance horizontal smart peaking	x	x	x
Digital Color Transient Improvement (DCTI)	x	x	x
Triple 10-bit Digital-to-Analog Converter (DAC)	x	x	x
Line locked PLL	x	x	x
Double window and picture-in-picture processing	x	x	
Embedded 80C51 microprocessor with 32 kB ROM and 512 Bytes RAM	x	x	x
I²C-bus controlled	x	x	x
Motion compensated upconversion of $1f_H$ video and film standards up to 292 active input lines per field	x	x	
Motion compensated upconversion of 50 Hz **video** or **film** standard (2:1) to 100 Hz (2:1)	x	x[1]	

Table 1: Features of IPQ modules

module:	1	2	3
Feature	**MK14-EM with SAA4998**	**MK14-EM with SAA4999**	**MK14-EM with SAA4999 (no PIP)**
Motion compensated upconversion of 50 Hz **video** standard (2:1) to 50 Hz progressive (1:1)	x	x	
Motion compensated upconversion of 50 Hz **film** standard (2:1) to 50 Hz progressive (1:1)	x		
Motion compensated upconversion of 60 Hz **video** standard (2:1) to 60 Hz progressive (1:1)	x	x	
Motion compensated upconversion of 60 Hz **film** standard (2:1) to 60 Hz progressive (1:1)	x		
3:2 pull-down motion compensation of 60 Hz film standard	x		
Full 8-bit accuracy	x	x	x
Edge dependent de-interlacing (EDDI)	x	x	
Variable vertical sharpness enhancement	x	x	
Motion compensated 3D dynamic noise reduction	x	x	
High quality vertical zoom	x	x	
On-board color decoder for main picture	x	x	x
On-board color decoder for subchannel (double window or PIP=picture-in-picture)	x	x	

Table 1: Features of IPQ modules

1. Reduced performance in film mode

3. Scan conversion overview

In this chapter an overview is given on the reasons for scan conversion as well as the various scan conversion modes offered by the Philips IPQ board. The modes can be divided into three main groups:

- scan rate doubling (50 Hz to 100 Hz or 60 Hz to 120 Hz)
- conversion to progressive scan (50 / 60 Hz interlace to 50 / 60 Hz progressive)
- non-integer scan rate conversion (50 Hz to 75 Hz, 60 Hz to 90 Hz, 50 Hz to 60 Hz)

3.1 Reasons for scan conversion

In standard TV systems (PAL, NTSC, SECAM) pictures are transmitted sequentially. The frequency is either 50 Hz (PAL, SECAM) or 60 Hz (NTSC). Originally it was defined according to the power line frequency of the respective countries in order to avoid interference. Both frequencies are so low that flickering is noticeable, with 50 Hz being clearly visible in large bright areas, while 60 Hz is not really annoying any more. “Flicker free” starts from 70 Hz on upwards.

For 50 Hz systems an increase in scan rate is therefore an essential picture improvement. Doubling the scan rate is technically the simplest approach and was therefore favored for many years. Now different conversion ratios like 50 to 75 Hz are also implemented.

The 60 Hz NTSC system has fewer lines than the 50 Hz systems (525 instead of 625), therefore line visibility, line flickering and less vertical resolution are more annoying than large area flickering. All these points can be effectively improved by conversion to progressive scan (non-interlace mode). This doubles the number of lines per field but leaves the field frequency unchanged.

3.2 Scan rate doubling

In scan rate doubling each input field in written into a memory and read from the memory at double the input clock and double line and field frequency. Advantageous are memories which allow writing and reading at the same time. On the other hand, random access is generally not required, the operation is more like a FIFO (first in - first out). Therefore these video memories do not need external addressing, address counters are internal. Just a reset is required on the input and output side to define the start of the field being written or read. Fig. 1 shows the block diagram of such a field memory.

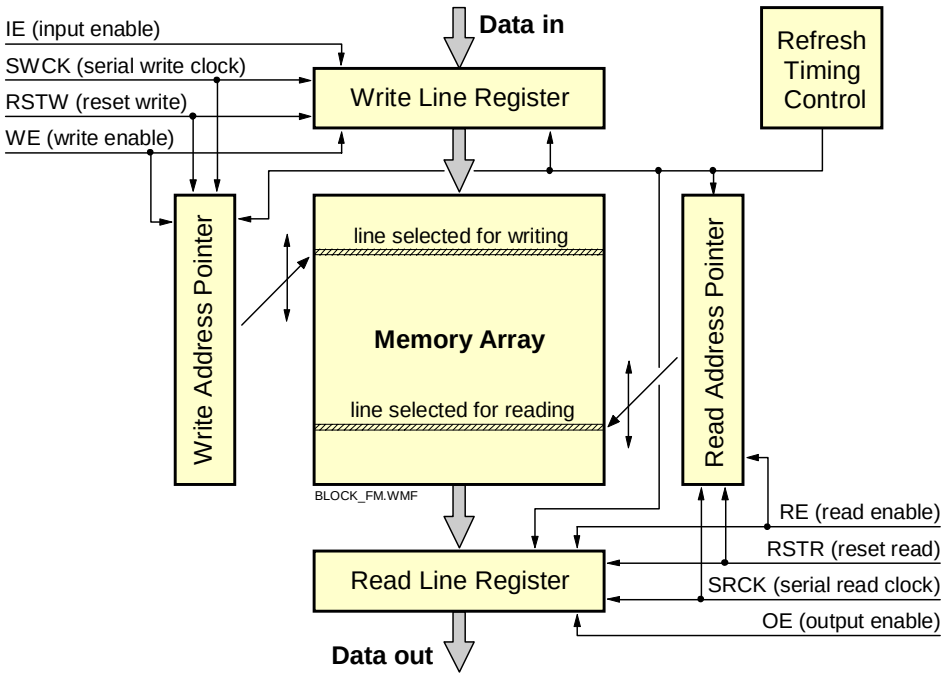


Fig. 1 Block diagram of the scan conversion field memory

3.3 Field repetition and frame repetition

TV pictures are transmitted in interlace format, each two fields make up one frame. The field frequency is 50 Hz (PAL, SECAM) or 60 Hz (NTSC), the frame rate thus 25 Hz or 30 Hz resp. In scan rate doubling two approaches are possible: field doubling or frame doubling. When designating the two fields as field A and field B then we can talk about the modes A-A-B-B (field repetition) and A-B-A-B (frame repetition). Fig. 2 shows the two modes for a simplified 9 line TV picture.

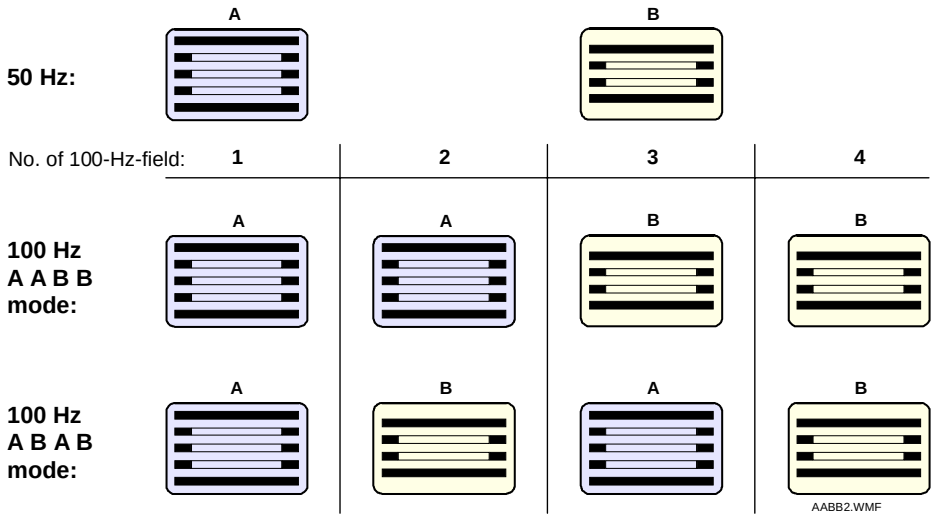


Fig. 2 Scan conversion modes A-A-B-B and A-B-A-B

In A-A-B-B mode interlace line flickering of 25 Hz (30 Hz) is still present, whereas in A-B-A-B mode it is practically not noticeable any more. With large area flickering being gone, the remaining 25 Hz line flickering in A-A-B-B mode is even more visible and annoying, therefore conversion to A-B-A-B mode is favorable because the line flicker frequency is doubled to 50 Hz.

3.4 Video mode and movie mode

In video mode (pictures taken by an electronic camera) every field represents another movement phase, so the rate is 50 (or 60 in NTSC) movements per second. We talk about movie mode (or film mode) if two consecutive fields are scanned from the same movie picture (running at 25 frames per second). In this case there is no movement between these two fields, the rate of movement phases is 25 per second.

If a movie is transmitted it is advantageous to choose A-B-A-B mode to reduce line flickering. In video mode however this sequence displays two movement phases in one frame generating contours and unsharpness around moving objects, therefore A-A-B-B mode is more suitable here.

3.5 Line flicker reduction (LFR)

In order to eliminate the line flickering for video sources and display these pictures in A-B-A-B mode also, a median filter is used. This filter has three inputs (pixel values) and outputs the median one, i. e. it discards the extreme values of the three. Fig. 3 show the basic principle.

This filter is used to generate the interpolated pictures in 100 Hz mode, see fig. 4. 100 Hz fields no. 1 and 4 are the original 50 Hz fields A and B, here the median filter is turned off. 100 Hz fields no. 2 and 3 have the median filter activated. In field 2 two of the filter inputs carry information from the neighboring lines of field A, one input has line information from field B. So picture content from field A dominates. This field is called

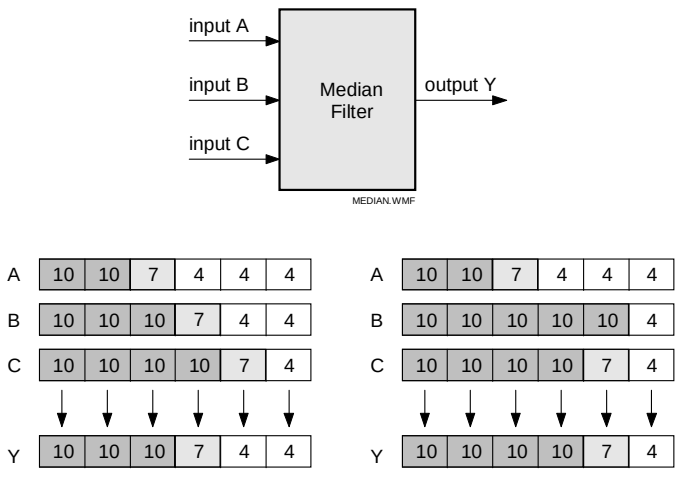


Fig. 3 Function of a median filter

A*. In 100 Hz field no. 3 two filter inputs have field B information and one has field A information, so the content of field B dominates. Whenever the line information from the non-dominating field fits in between the lines of the dominating field, this line is output. In all other cases the median value is that of the dominating field. These decisions are taken on a pixel-by-pixel basis, so the filter can be seen as switch between an A-A-B-B and A-B-A-B display on a pixel-by-pixel basis, depending on the picture content.

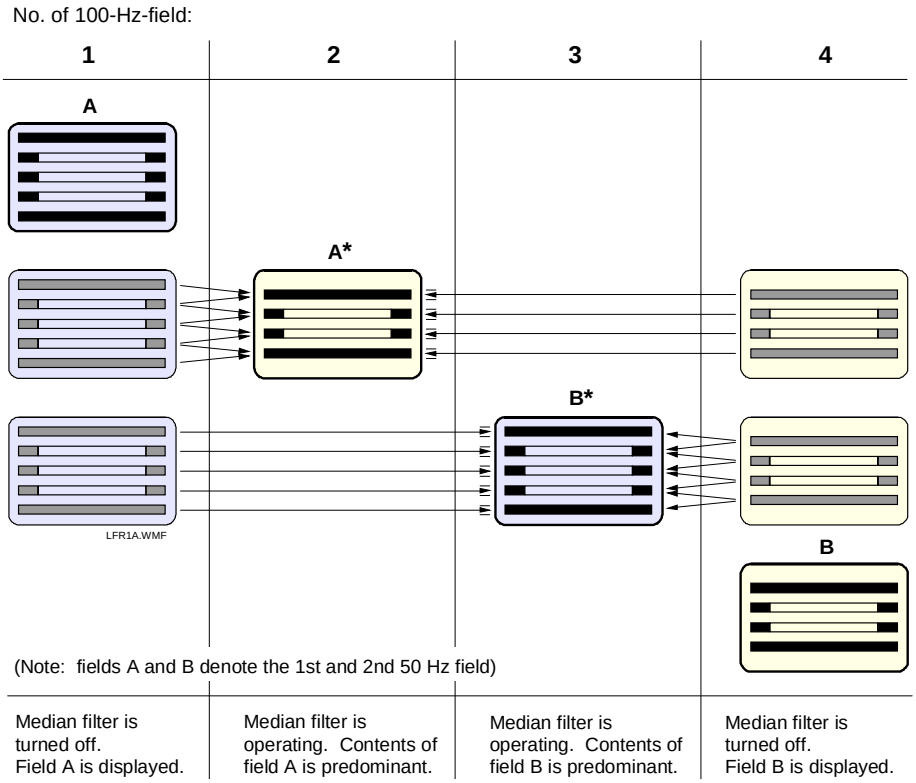


Fig. 4 Median filter used to generate interpolated pictures

The above described line flicker reduction mode is also known as ***Digital Scan*** in Philips IPQ concepts.

3.6 Display of moving objects

When a movie (25 movements per second) is viewed on a 50 Hz TV screen, moving objects do not move smoothly but show a judder or double contour. This is due to the fact that the rate of movement (25 Hz) is not equal to the display rate (50 Hz), see fig. 5 center. A similar effect occurs when the display rate is increased to 100 Hz. With scenes transmitted in video mode (50 Hz) contours of moving objects appear double. When a movie (25 Hz) is displayed on a 100 Hz screen contours appear even four times, see fig. 5 bottom.

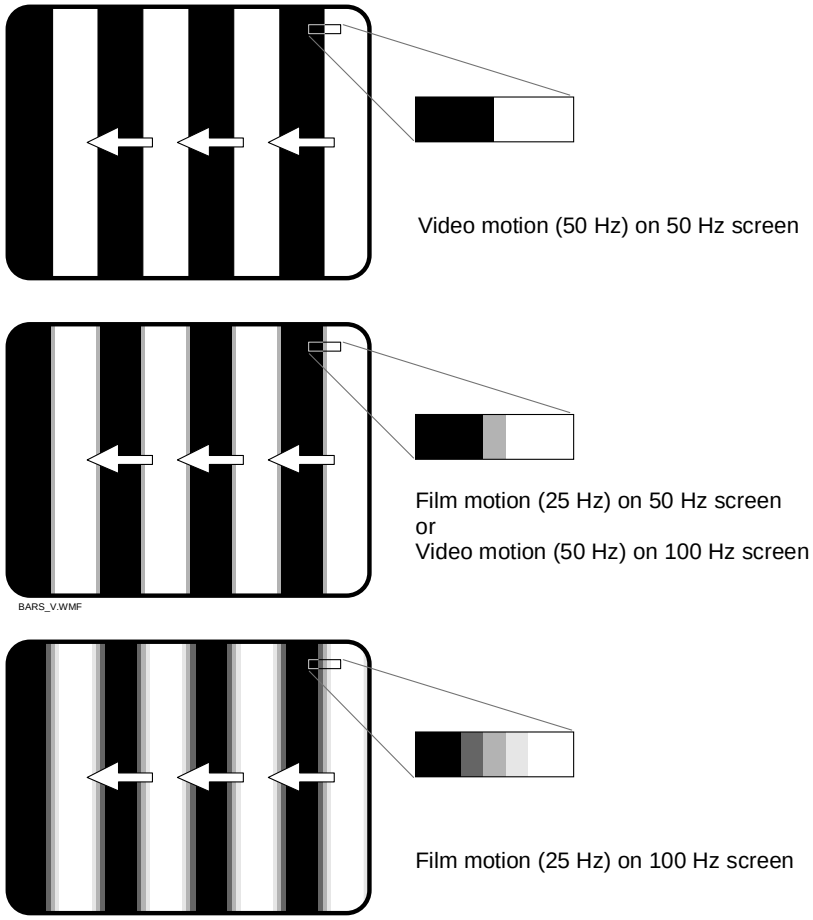


Fig. 5 Moving bars: motion artefacts along the edges

In a diagram depicting the motion of objects over time (fig. 6) the reasons for the double or multiple contours become more clear: When fields are simply repeated each object appears twice (or four times) at the same location, then jumps to next location. A viewer's eye tracing the object moves uniformly along the track and notices the off-track repetitions as contours.

In fig. 6 the upper two diagrams show video and movie sources on a 50 Hz display, here the video source will appear sharp, the movie source will be juddering due to the low-frequent (25 Hz) field repetition. The next two diagrams show the same two sources converted to 100 Hz, here the video material will show a double contour and the movie material a fourfold contour. In the lower three diagrams motion compensation is applied to the upconverted signal. The third row depicts what was possible with Philips' first generation motion compensation IC, the SAA4991 (MELZONIC). Objects in the interpolated picture are placed in the middle between the two original positions of the actual and previous field. For movie sources this means that a remaining judder or contour is still there, because the motion rate is converted from 25 Hz to only 50 Hz, while the display rate is 100 Hz. The successor ICs SAA4992/3/4 (FALCONIC, FALCONIC+, RAVEN) do better in this respect, they are able to convert movement from 25 Hz to 100 Hz and thus correctly position objects even in movie mode.

3.7 2:2 and 3:2 pull-down movie modes

Movies are recorded on film at 24 frames per second (fps). For transmission in a PAL or SECAM TV system (50 Hz) the frame rate is increased slightly to 25 fps, and each frame is scanned twice to generate the two interlaced fields for TV. This mode is called 2:2 pull-down, see fig. 7.

For transmission in the NTSC TV system (60 Hz) a film is run at its original 24 fps. Then one frame is scanned twice and the next one three times, this gives five fields for every two film frames and a field frequency of 60 Hz. This mode is called 3:2 pull-down, see fig. 8.

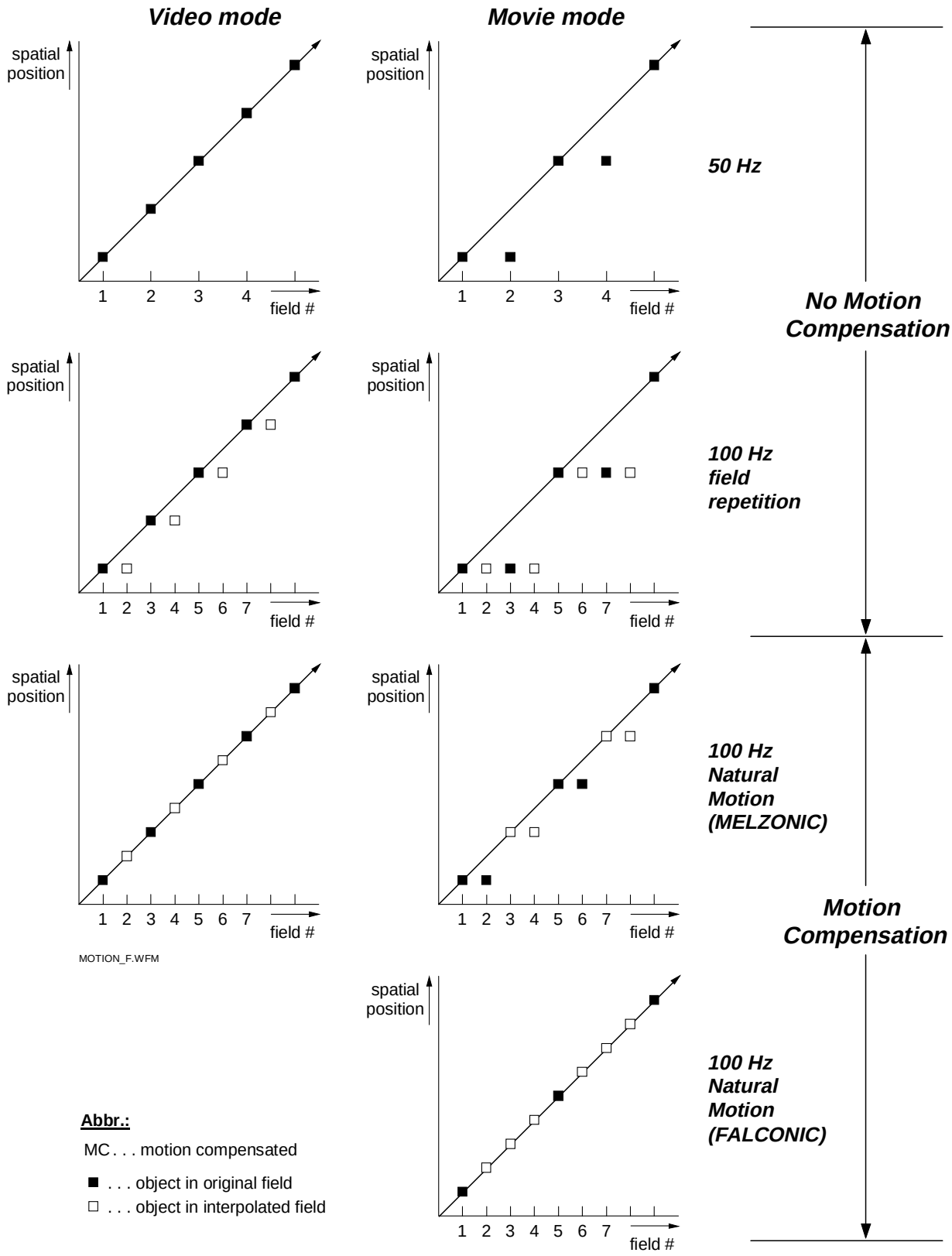


Fig. 6 Scan conversion with and without motion compensation

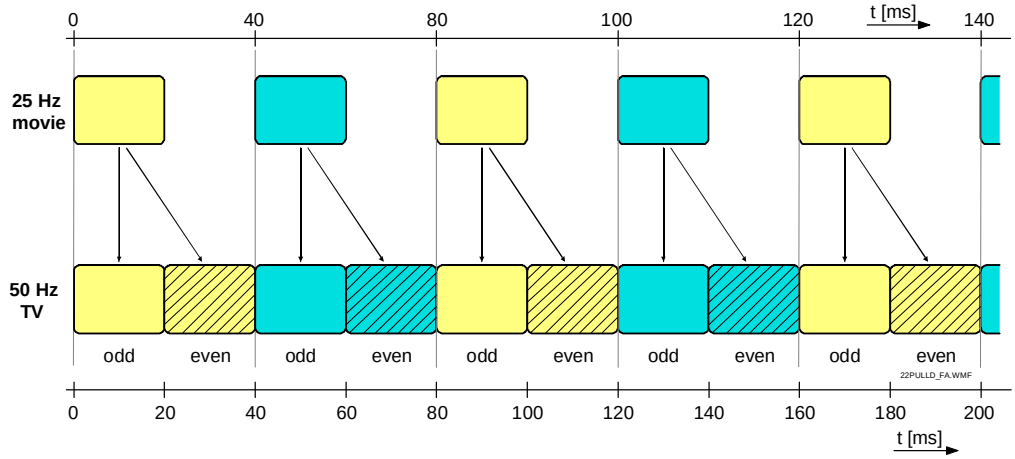


Fig. 7 2:2 pull-down movie move

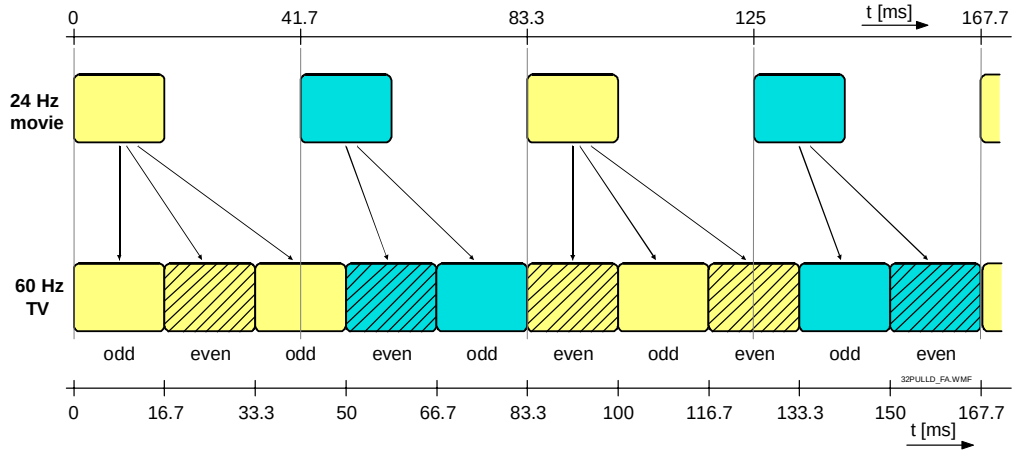


Fig. 8 3:2 pull-down movie move

3.8 Motion compensation in movie mode

Motion estimation and compensation in movie mode is in certain ways different from video mode:

- The rate of movement is lower. Due to the frame rate of 24 or 25 frames per second the motion displacement between movement phases is about twice as large as it is in video mode with 50 or 60 movements per second. This means that the movement vector for a comparable motion is also about twice as large. After scene changes it is therefore more difficult for the estimation algorithm to set up the new vector field, and it takes somewhat longer due to the lower movement update rate.
- The proper movement phase must be determined. In the incoming video from a movie source any two consecutive fields which are taken from the same piece of film do not contain motion. Motion can only be detected between every other pair of fields which are from different film frames. It is important to determine the correct phase and combine the right two fields, because a wrong phase will deteriorate the motion rendition in the display.
- In 3:2 pull-down mode phase detection in movie mode is still more difficult. There are alternately 3 and then 2 consecutive fields without motion, and this phase has to be detected reliably.
- During motion compensation the vectors found during estimation are split into different fractions. Scan rate doubling In video mode means that the interpolated field lies in-between the two original ones, so 50% of the displacement vector has to be applied for the new field. In movie mode there is only one original field and 3 interpolated ones, with an applied motion vector of 25%, 50% and 75% respectively, see fig. 9. When applying motion compensation while converting NTSC 3:2 pull-down movie to 60 Hz progressive scan, the applied motion vectors are 40%, 80%, 20% and 60% respectively, see fig. 10.

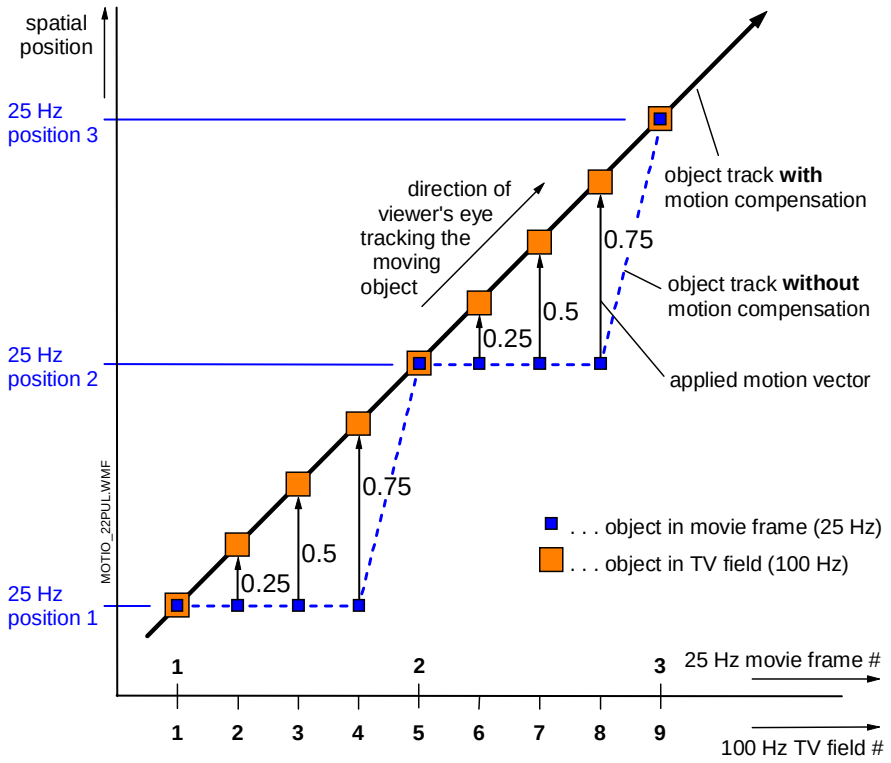


Fig. 9 Motion compensation in 2:2 pull-down movie mode

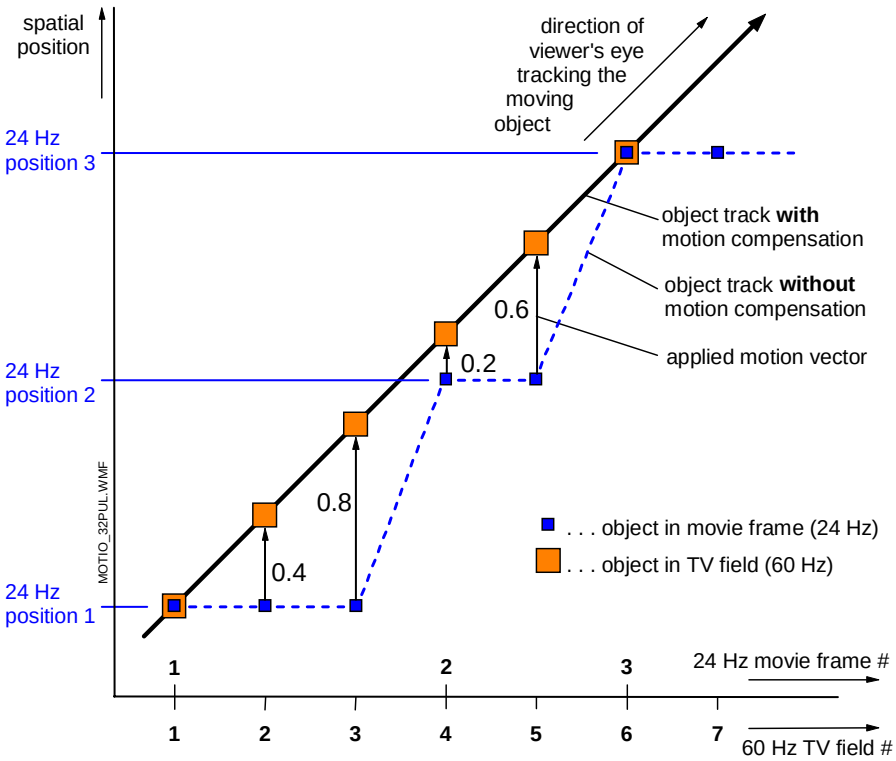


Fig. 10 Motion compensation in 3:2 pull-down movie mode

3.9 Progressive scan

Especially in the NTSC standard (60 Hz), where flicker reduction is not a main feature, picture improvement by doubling the line number is preferred. This mode is called *progressive scan (proscan)* or *non-interlace*. In the 50 Hz standard this mode is possible also. The improvements are:

- Line flickering is gone. Line flickering appears along horizontal structures (transients in vertical direction) which tend to flicker with the interlace frequency of 25 Hz. This low frequency is clearly perceptible and therefore annoying.
- Line structure is gone. No more single lines are visible even from close viewing distance from the screen. The picture appears clear and smooth.
- Line crawling is gone. This effect can happen at very close viewing distance from the screen when the viewer sees the line structure moving up or down with a fixed speed, especially in areas without much detail. This effect is generated by the interlaced display. The crawling speed is 11..12 sec. per screen height in PAL and 8 sec. per screen height in NTSC.

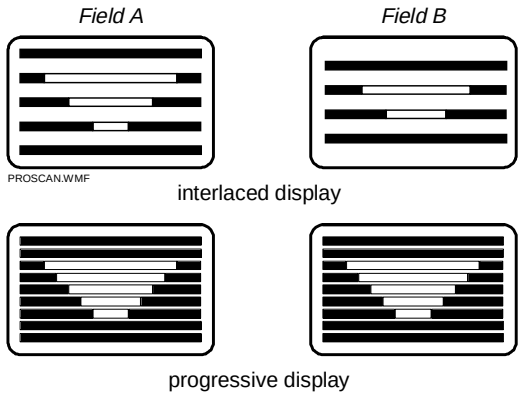


Fig. 11 Interlaced display vs. progressive display

Fig. 11 shows how two interlaced fields are combined and displayed in non-interlace (progressive) mode. If the input video is from a film source (no motion between two fields of a frame) then a field merge is done. In all other cases the median filter is active selecting the fields on a pixel-by-pixel basis.

Just like in 100 Hz, in progressive scan mode the line frequency is double the input line frequency. So compared to 100 Hz which is generally displayed in interlace, only the field frequency is different. TV display units can fairly easily adapt to 50 (60) Hz or 100 Hz, therefore it is possible to display PAL in 100 Hz interlace and switch to 60 Hz progressive for NTSC program sources.

3.10 75 Hz interlace

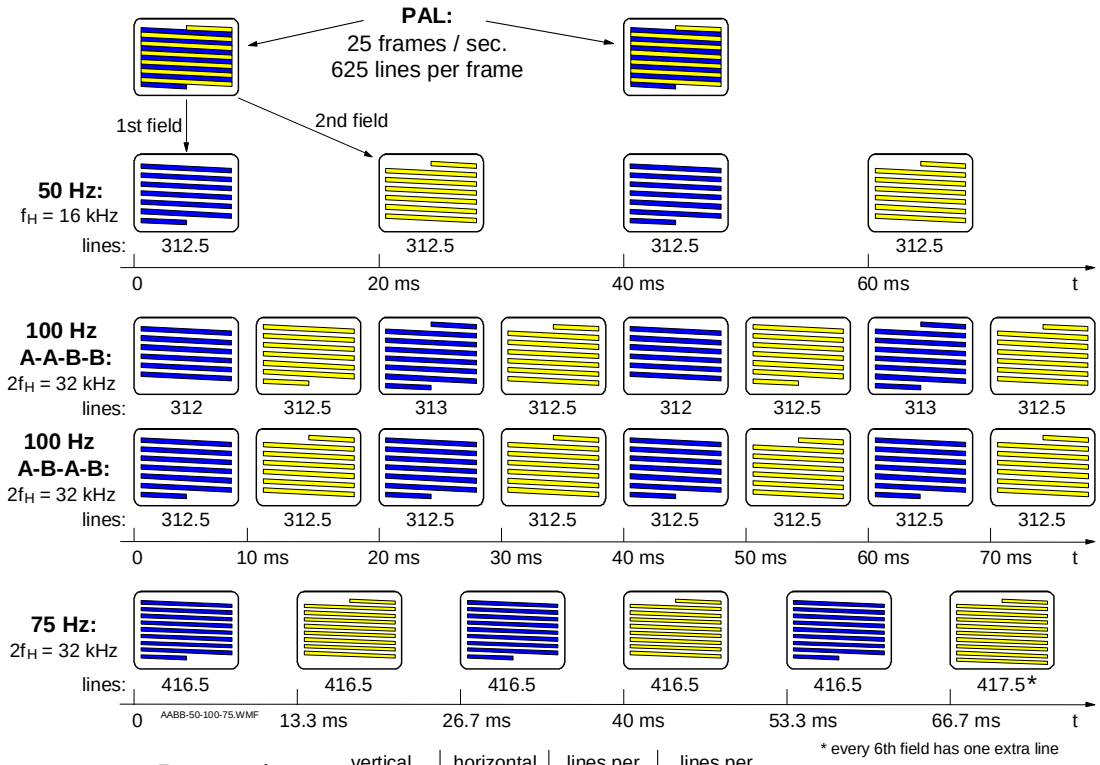
As said in the beginning of this chapter, field scan rates above 70 Hz are considered “flicker free”. So scan rate doubling to 100 or 120 Hz is beyond of what is necessary to eliminate flickering. But a conversion factor of 2 was easy to realize when scan conversion first came up.

Doubling the field scan rate requires to also double the horizontal scan rate and to double the analog bandwidth of the amplifier stages of the video signal when it is to be displayed on the screen with the same sharpness compared to 50 or 60 Hz. But now that these problems are solved different use can be made of these developments: with practically no change to the deflection and amplifier stages the number of visible lines (and thus pixels) can be increased by reducing the scan rate from 100 Hz to 75 Hz, a rate which also is sufficient to have a flicker free picture. The number of lines can be increased from 625 to 833 while staying with the same horizontal deflection frequency. This mode is called “75i” and is implemented in the latest version of the SAA4979’s firmware. Its advantages are:

- considerably less line visibility and line flickering
- increased vertical sharpness, if appropriate vertical peaking and signal processing is applied
- no changes in hardware, the mode runs with all 100 Hz deflection units

Motion compensation is also available and ensures smooth movements in video and movie mode comparable to those in 100 Hz.

Fig. 12 shows a comparison of the scan modes 50 Hz, 100 Hz and 75 Hz. In 75 Hz every 6th field has one extra line to make the conversion come out even:



For comparison:

vertical frequency	horizontal frequency	lines per field	lines per frame
50 Hz	16 kHz	312.5	625
100 Hz	32 kHz	312.5	625
75 Hz	32 kHz	416.5	833

Fig. 12 Format comparison 50 Hz / 100 Hz / 75 Hz

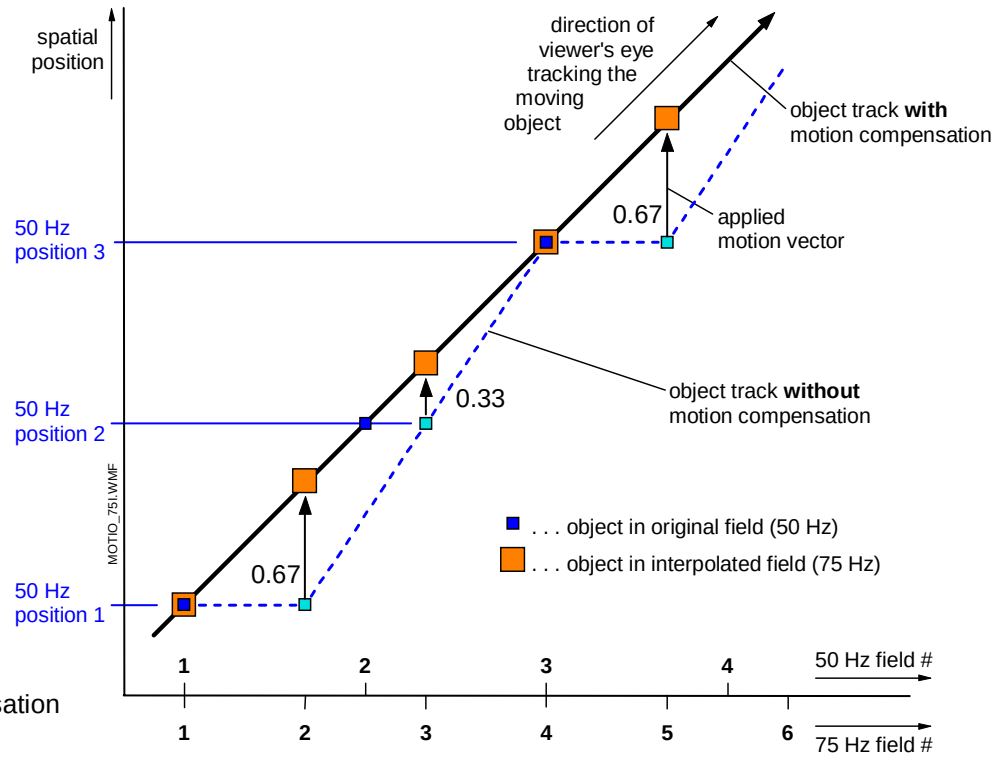


Fig. 13 Motion compensation in 75i mode

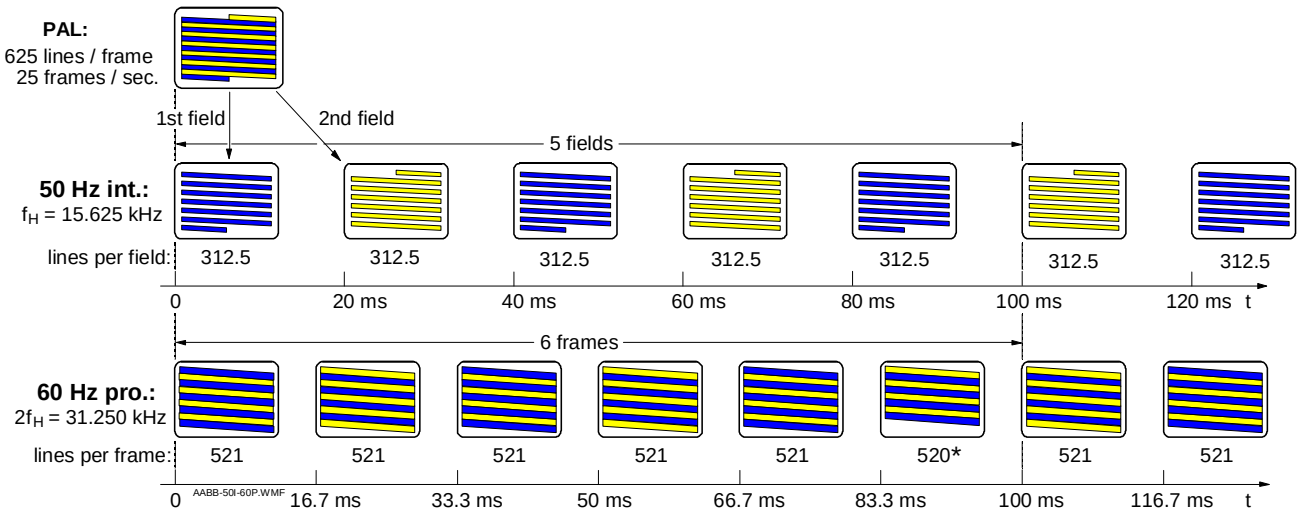
100 Hz: 8 fields x 312.5 lines = 2500 lines
75 Hz: 6 fields x 416.5 lines + 1 line = 2500 lines

In order to get a stable picture in vertical scan direction, the display needs a DC-coupled deflection unit. But this is required for 100 Hz too, if the display runs in field-doubling mode (AABB) where fields are of unequal length also (312 - 312.5 - 313 - 312.5 lines)

3.11 Conversion from 50 Hz to 60 Hz

In order to support applications requiring 60 Hz vertical frequency at the output, a conversion from 50 Hz to 60 Hz has been incorporated into the software. 60 Hz is not quite flickerfree, but much better than 50 Hz in that respect. Especially when it comes to driving flat panel displays 60 Hz can be favorable.

Conversion is done from 50 Hz interlace to 60 Hz progressive. This output format also runs on double the input line frequency ($2f_H$ or 31.250 kHz), just like 100 Hz or 75 Hz interlace. Because it is derived from the 50 Hz PAL input signal ($f_{H\text{-}PAL}$ = 15.625 kHz) which differs from the NTSC line frequency ($f_{H\text{-}NTSC}$ = 15.750 kHz) the number of lines per frames is slightly different from the standard 525 of the NTSC system. Five 50 Hz input fields (interlace) are converted to six 60 Hz output frames (progressive), five of them having 520 lines and the sixth one 521 lines, see fig. 14. Like the other display mode with unequal field/frame length sequences (100 Hz AABB mode or 75i), it is required to use a DC-coupled deflection stage for the CRT otherwise a vertical jitter can occur.



For comparison:

vertical frequency	horizontal frequency	lines per field	lines per frame
50 Hz (int.)	15.625 kHz	312.5	625
60 Hz (pro.)	31.250 kHz		520 or 521*

* every 6th frame has one line less

Fig. 14 Scan conversion from 50 Hz interlace to 60 Hz progressive

While converting from 50 Hz to 60 Hz basically every 5th 50 Hz field is repeated to get 60 Hz. When watching moving objects these would show a low-frequent and rather annoying judder of 12 Hz. But in this mode also motion compensation is active. Motion vectors are determined and moving objects are shifted toward the position expected by the viewer's eye tracking the moving object, see fig. 15. A suitable fraction of the vector is applied depending on the current phase between the 50 Hz input and the 60 Hz output signal.

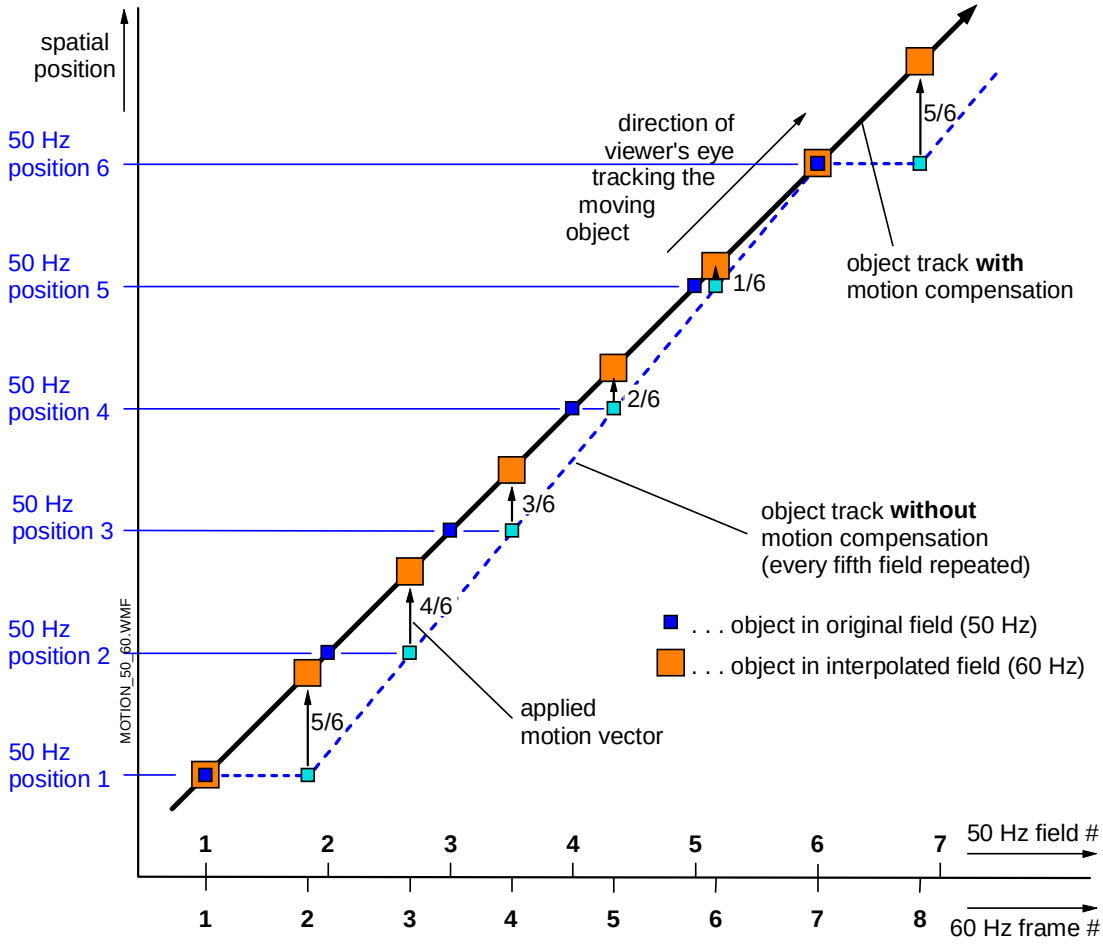


Fig. 15 Motion compensation in 50i/60p conversion mode

3.12 Double clock system

In our scan conversion concepts the display line frequency and system clock are double that of the acquisition side. In a simple approach a PLL could generate the display clock from the input line frequency, with the acquisition clock being just derived from the display clock by a division of 2. In fig. 16 this is shown by the dotted line.

This approach has the disadvantage that an unstable horizontal sync pulse as it is generated for example by a VCR (especially during head change) will influence both the acquisition clock and the display clock. An improvement is to use two PLLs and to optimize each one separately: the acquisition PLL is to be made fast, so it can easily follow any time base changes of the input sync; the display PLL is to be made slow in order to generate a stable display.

The SAA4979 has the display PLL integrated onto the chip. The horizontal reference pulses are taken from the ITU data stream. The acquisition PLL is located in the color decoder (SAA7114/..15/..18).

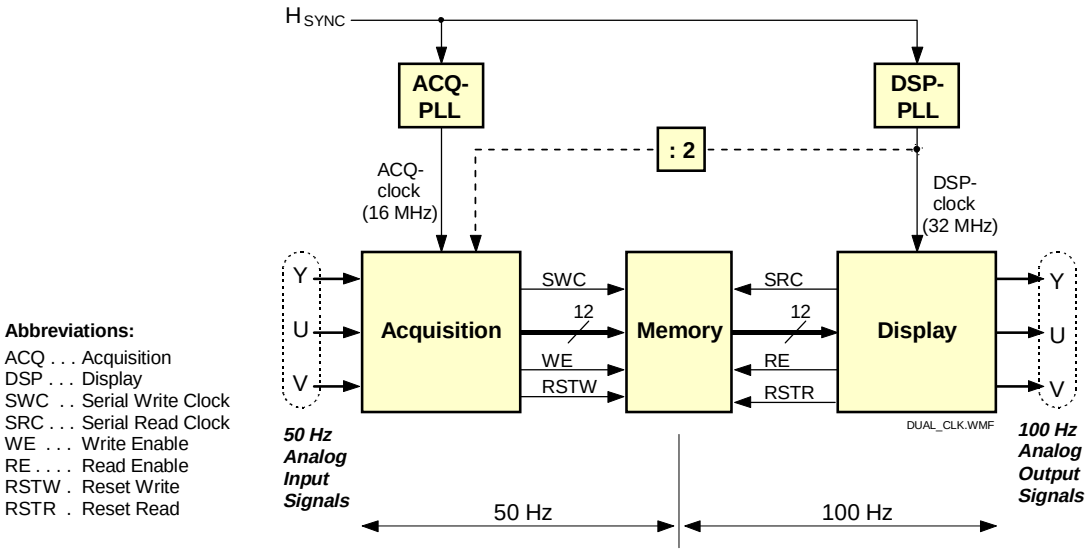


Fig. 16 Single clock and dual clock system in scan conversion

4. Functional description of the SAA4979

The SAA4979 is a stand-alone IC for 4:2:2 video scan conversion. Conversion can either be field rate doubling from 50/60 Hz to 100/120 Hz or conversion to progressive scan. The main characteristic is that all digital functions including a 3.5 MBit field memory are placed inside one IC.

The IC supports two digital ITU-656 video input data streams to allow picture-in-picture processing. It provides picture improvement features and non-linear horizontal picture compression or expansion and has analog YUV outputs for a display. The on-chip memory is used for scan conversion as well as for field-based noise reduction. Various video enhancing features are provided which are controlled by the embedded 80C51 microprocessor core. An I²C bus interface offers communication with an external master controller.

The SAA4979 is designed especially for an economy 100 Hz application and allows a one-chip 100 Hz concept. It is the successor of the SAA4977. For mid- and high-end applications it offers an expansion port for vector based motion estimation and compensation ICs like the SAA4992/3/4 (FALCONIC) or the SAA4998 (FALCON-IC-EM). Besides motion-compensated field rate up-conversion, these ICs offer de-interlacing, noise reduction and zoom.

The functional block diagram of the SAA4979 is shown in fig. 17. The IC is made up of two chips (multi-chip module, MCM). The horizontal dashed line marks the border between the two dies: the upper part contains the front end processing and the 3.5 MBit embedded DRAM for field storage. The lower part contains further signal processing and the backend of the IC. The shaded area marks the part which runs with basic line and field frequencies and a clock of 27 MHz for the ITU656 coded input signals. The non-shaded part runs with double line and field frequency and a clock of 32 MHz. Throughout the IC signal processing in the 4:2:2 format is performed.

4.1 Digital processing at $1f_H$ level

4.1.1 ITU-656 decoder

The SAA4979H is equipped with two digital inputs for 8 bit wide Y/UV signals in the 4:2:2 format complying to the ITU-656 standard. There are two separate decoders with equal functions. Decoder 1 gets the main input signal and decoder 2 gets the second signal, so two signal sources can be displayed on the screen at the same time, e.g. as DW (double window), PIP (picture-in-picture) or POP (picture-outside-picture, on a side panel).

Data is input at 27 MHz with Y and U/V samples alternating in the following order: U0 - Y0 - V0 - Y1 - ... , see also fig. 18. 720 pixels are processed per active video line with timing reference codes being inserted at the beginning and end of each line. A 'Start of Active Video' (SAV) code is generated before the first active video sample and a 'End of Active Video' (EAV) code after the last active video sample, see fig. 19 where the position of horizontal timing reference codes is depicted.

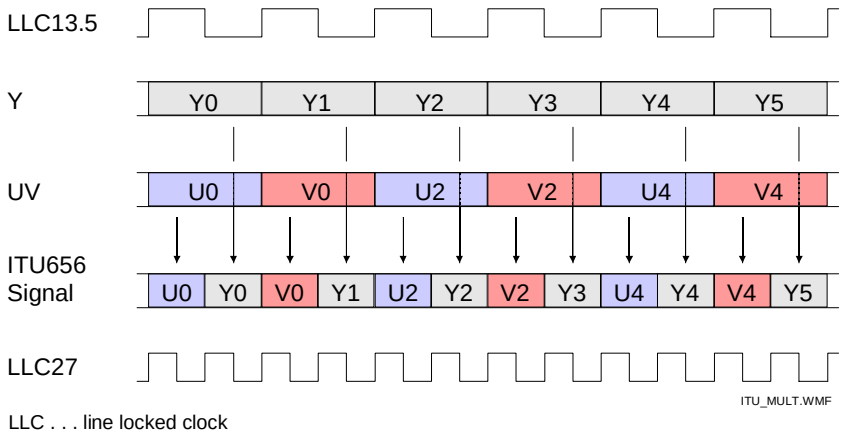


Fig. 18 ITU-656 multiplex signal

The active video data words are limited to 1 to 254, since the data words 00_H and FF_H are used for identification of the timing reference codes. These codes are packets of four data words with the first three being FF_H - 00 - 00. The fourth word has bits identifying the beginning and end of horizontal and vertical blanking as well as the first and second field. An overview of the video timing reference codes is given in fig. 20.

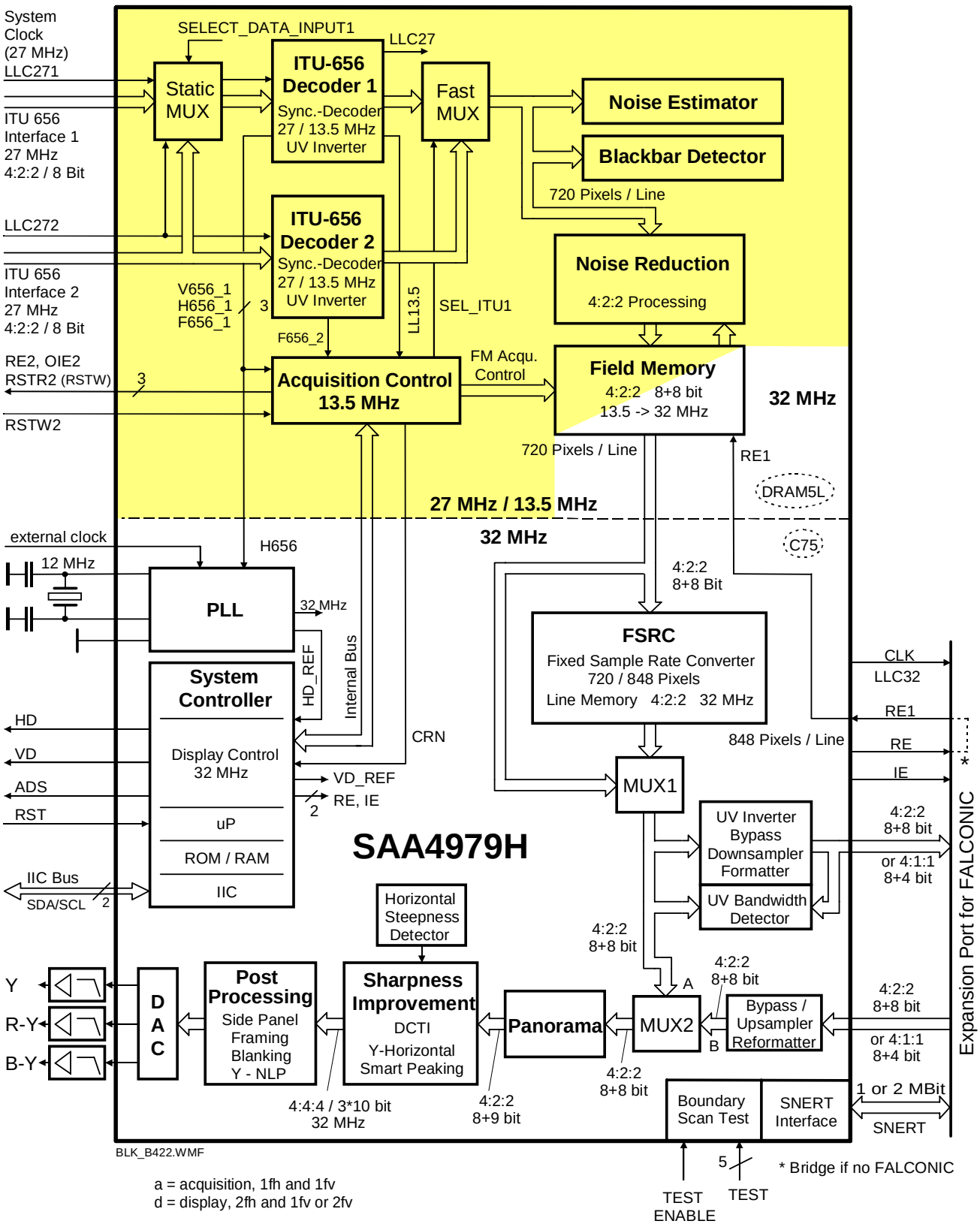
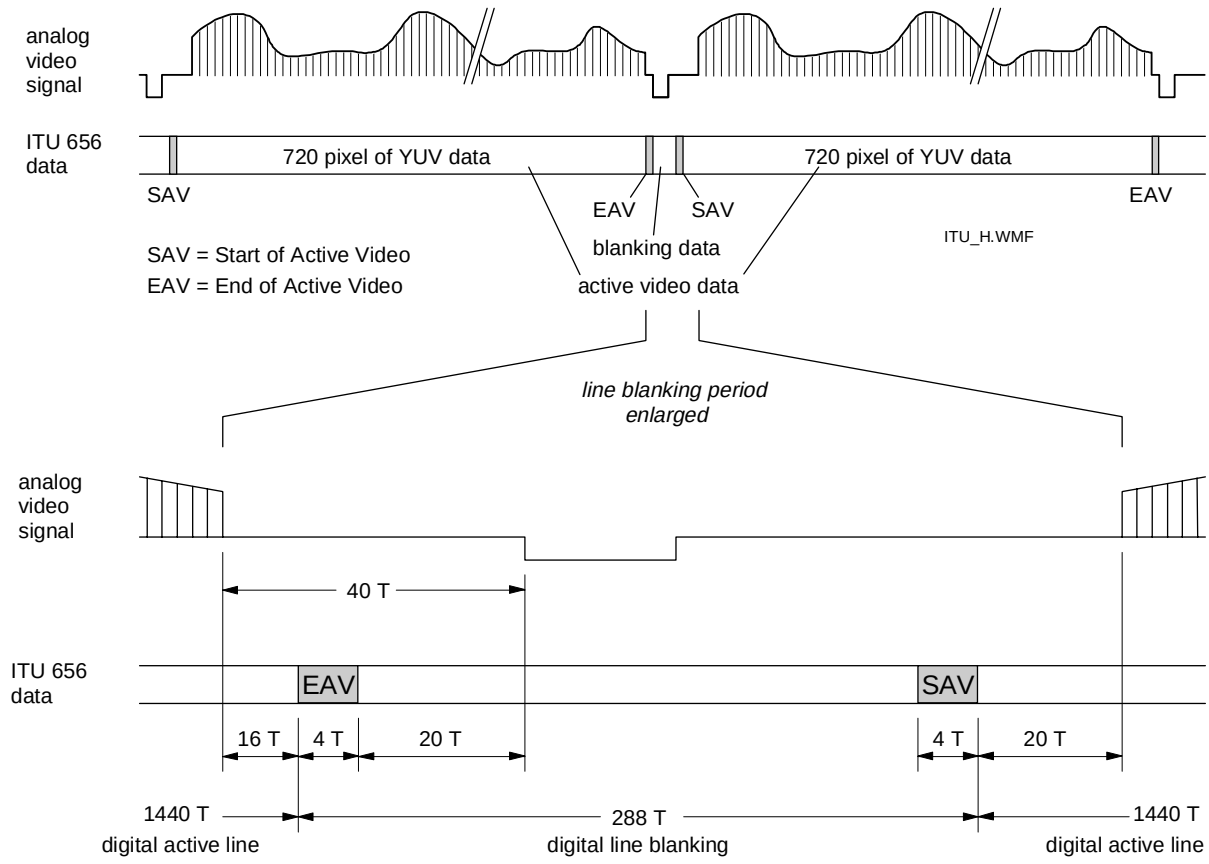


Fig. 17 Block diagram of the SAA4979H



LLC27 = 27 MHz

T = 1/LLC27 = 37 ns

digital line = digital line blanking + digial active video = 1728T

Fig. 19 ITU-656 horizontal timing

Data bit number	First word (FF)	Second word (00)	Third word (00)	Fourth word (xy)
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P_3
2	1	0	0	P_2
1	1	0	0	P_1
0 (LSB)	1	0	0	P_0

ITU_CODE.WMF

F = 0 during field 1
= 1 during field 2

V = 0 elsewhere
= 1 during field blanking

H = 0 in SAV (Start of Active Video)
= 1 in EAV (End of Active Video)

P_0 ... P_3: protection bits for 1-error correction and 2-error detection

Fig. 20 ITU-656 timing reference codes

While bit H in the fourth code word denotes the beginning and end of the active video line, bit V defines the beginning and end of vertical blanking and bit F serves for identifying first and second field. The lines numbers where these bits change are given in fig. Fig. 21.

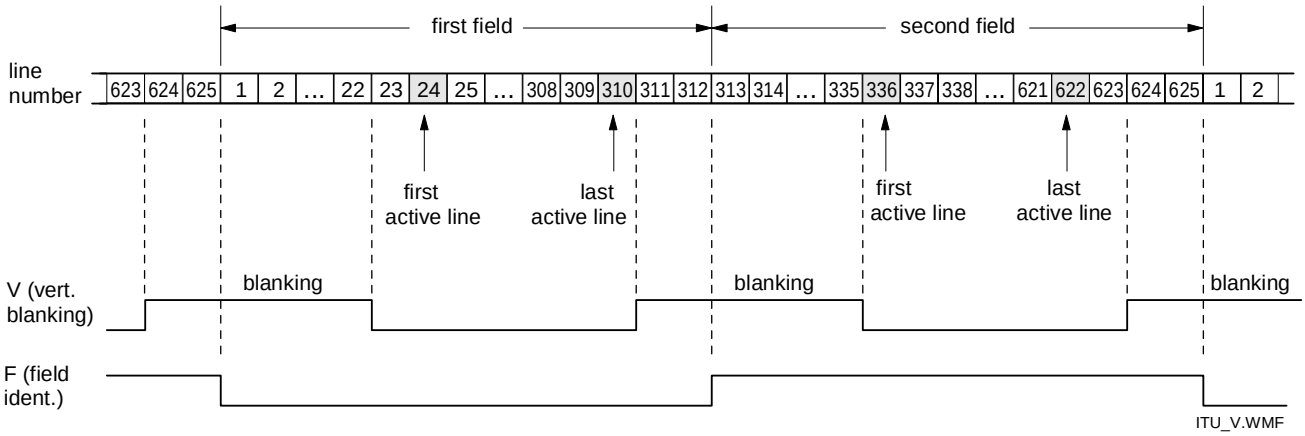


Fig. 21 ITU-656 vertical timing

4.1.2 Inputs

There are two inputs for digital video data and two ITU-656 decoders. Each 8-bit wide data channel has its own 27 MHz line-locked clock LLC27. One of these inputs can be selected for the ITU-656 DECODER 1 by the STATIC MUX in front of it. Alternatively both inputs can be fed to both ITU-656 decoders and the signals are combined by the FAST MUX to a double window or to a picture-in-picture (PIP) image. Only decoder 1 provides pulses for vertical, field, horizontal and clock synchronization to the ACQUISITION CONTROL block and to the PLL while decoder 2 delivers only field phase information.

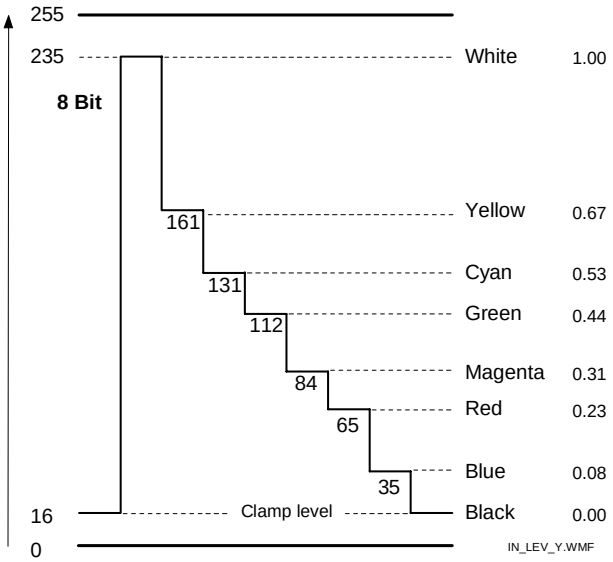


Fig. 22 Digital levels of Y input signal for color bar 100/0/75/0 (ITU-601)

The levels of the input signal components Y, Cr, Cb (”YUV”) are depicted in fig. 22, 23, and 24, the right most scale is normalized to the amplitude 1.

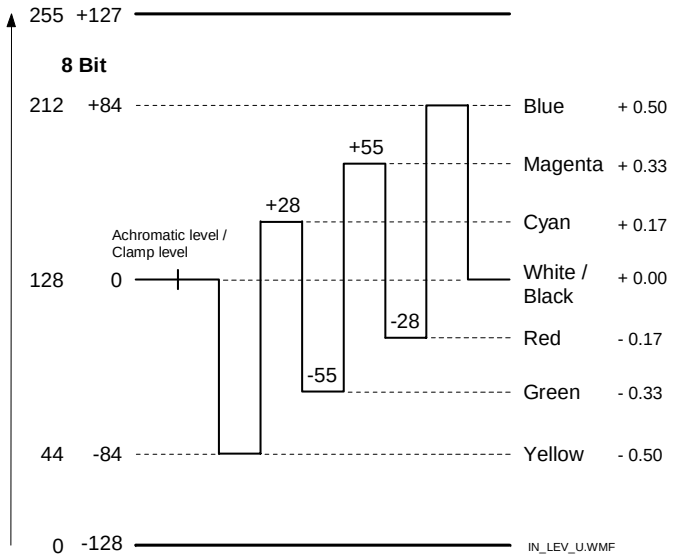


Fig. 23 Digital levels of U input signal for color bar 100/0/75/0 (ITU-601)

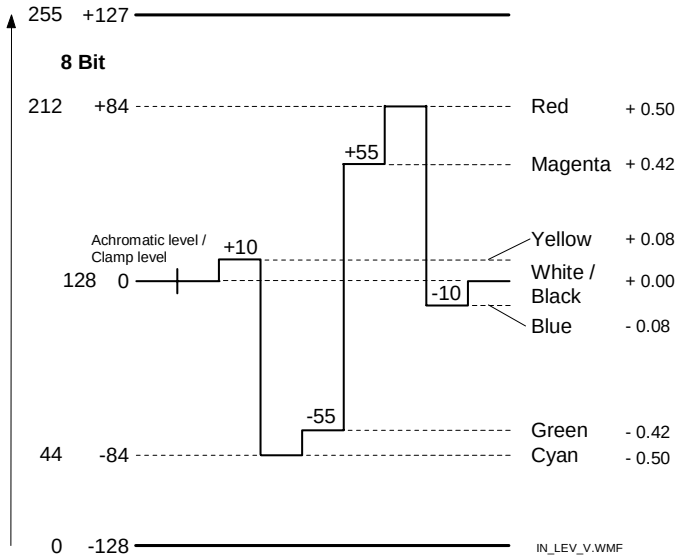


Fig. 24 Digital levels of V input signal for color bar 100/0/75/0 (ITU-601)

4.1.3 Double window and picture-in-picture processing

Data from the subchannel can be inserted into the data stream of the main channel by means of a fast switch. The two channels can be used together with one or two external field memories to implement e.g. double window or picture-in-picture (PIP) processing. In case of PIP processing the second input has to be scaled to the desired size, in case of double window also the main input has to be scaled. This scaling has to be done in front of the SAA4979, e. g. in the color decoder or special buffer memories which in case of PIP are also needed for synchronization. This synchronization of the subchannel to the main channel is achieved by providing synchronized read signals (RE2 and RSTR2) for the external field memories, whereas the write signals need to be provided together with the incoming data by the external signal source. Both field based and frame based PIP processing is supported. Also a multi-PIP mode is possible by freezing the data in the internal field memory within certain areas via the programmable internal control signal IEint.

4.1.4 Black bar detector

Wide screen transmissions ("letterbox format") displayed on a conventional 4:3 screen will result in black bars at the top and bottom, see ⓐ in fig. 25. If no measures are taken, they will also leave these black bars on a wide

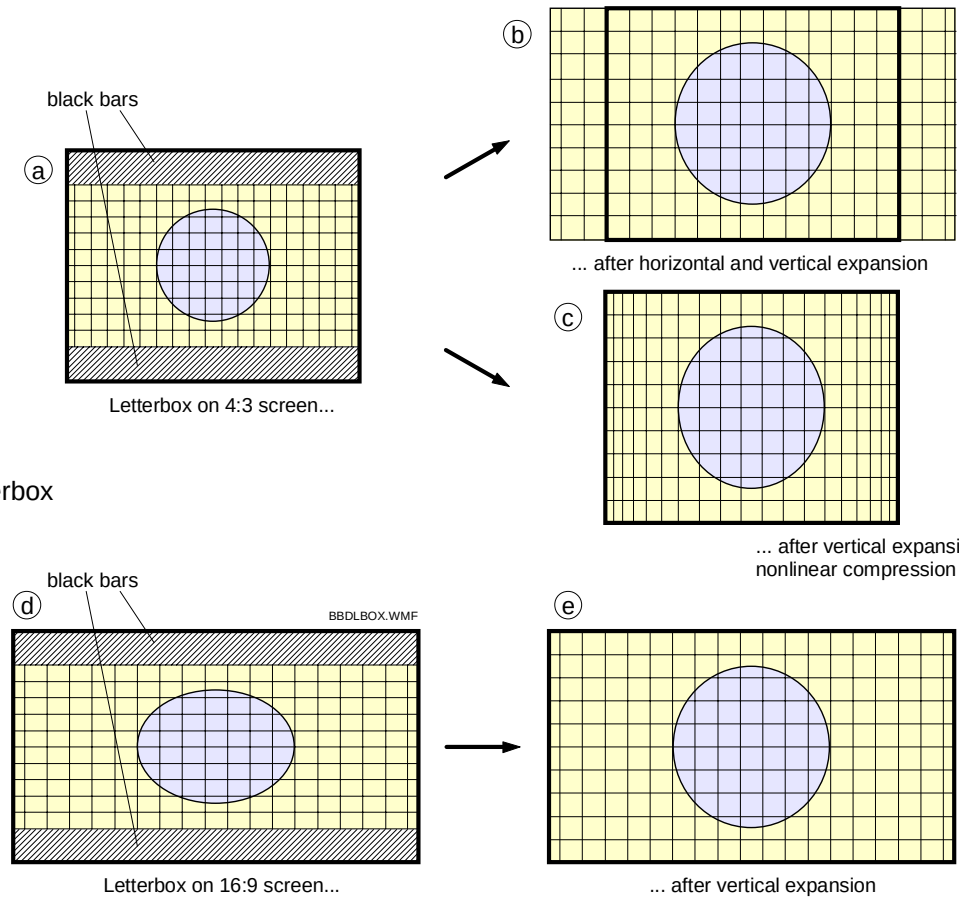


Fig. 25 Dealing with letterbox transmissions

16:9 screen, see ⓓ in fig. 25. On a 4:3 screen this is acceptable because the proper aspect ratio is maintained. On a 16:9 screen it appears distorted however, which is less acceptable. In fig. 25 some measures are pointed out of what to do with a letterbox signal.

On a 4:3 display the picture can be expanded horizontally and vertically to fill the whole screen, this results in some parts of the picture getting lost (left and right side, see part ⓑ in fig. 25). Another way is to expand the picture vertically and activate the inverse panorama mode (“amaronap mode”) which compresses the left and right side so everything fits on the screen (ⓒ in fig. 25).

On a 16:9 screen the only desirable action would be to expand the picture vertically. This would fill the whole screen and restore a proper aspect ratio (ⓔ in fig. 25).

When there is no information transmitted about the picture format, the display has to be adjusted manually. An automatic mode though becomes available if the blackbar detection of the SAA4979 is activated and its results are evaluated.

Fig. 26 shows the block diagram of the black bar detection. All measurements are done in a rectangular window which is defined by the four parameters *BBD_HSTART*, *BBD_HSTOP*, *BBD_VSTART*, and *BBD_VSTOP*. The horizontal start and stop position can be programmed in steps of 4 pixels, the vertical position in steps of one line.

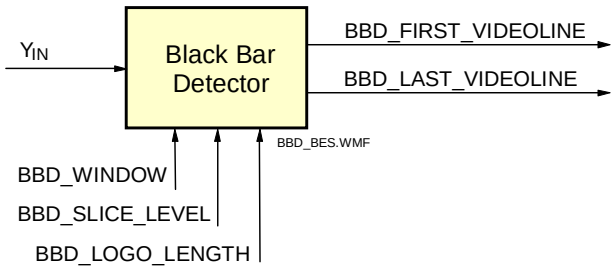


Fig. 26 Block diagram of the black bar detection

The aim of the black bar detector is to determine the first and last non-black line in the picture. At the beginning of a field a temporary register *first_videoline* is incremented every line as long as the line is found to be black. The incrementing stops with the first non-black line. This one represents the top of the letterbox picture. The register content can be read as *BBD_1ST_VIDEOLINE*. The bottom of the letterbox picture is found in a similar way. Incrementing of the temporary register *last_videoline* stops with the last non-black line, and the register content can be read as *BBD_LAST_VIDEOLINE*. *BBD_1ST_VIDEOLINE* is a 7-bit value and *BBD_LAST_VIDEOLINE* is an 9-bit value. Both read registers of the black bar detector can be read by the internal microprocessor only. They are evaluated and the result is available in the read register 09_H.

Recognizing a black line can be influenced not only by adjusting the window borders, but also by two more parameters. *BBD_SLICE_LEVEL* determines whether a pixel is considered black or not and *BBD_EVENT_VALUE* sets a limit on how many non-black pixels are allowed while that line is still considered to be black. Both parameters are entered as 6-bit values which are internally multiplied by 2 to get the actual slicing level or event number.

4.1.5 Dynamic noise reduction

The dynamic noise reduction circuit in the SAA4979 is based on a recursive signal filtering in which an actual and a previous (field delayed) signal are mixed. The level of the noise reduction is dynamically controlled depending on movement, i. e. depending on differences between pictures. The circuit therefore is closely related to the IC's field memory. This memory has two output ports: one is used for double scan rate and the second one is a 50/60 Hz output and is used for the noise reduction loop. Fig. 27 shows the block diagram of the noise reduction circuit. A more detailed diagram can be found in the data sheet of the SAA4979[1].

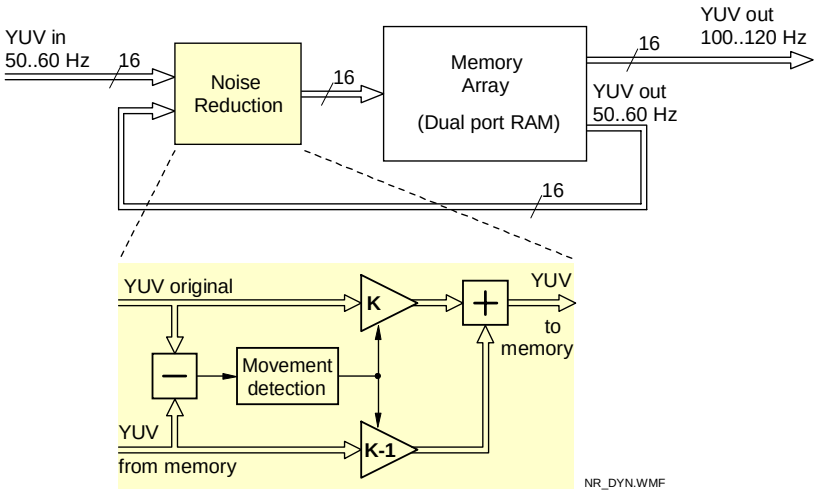


Fig. 27 Basic block diagram of the DNR circuit

Noise reduction can be activated by forcing the *NREN* control bit to HIGH. The amount of noise reduction is controlled by the k-factor. K determines the part of fresh video information and is between 0 and 1. A low value of k

1. SAA4979H: Philips Semiconductors data sheet

means a high amount of recursion (and thus high noise reduction) while for k = 1 noise reduction is turned off. In 'fixed k mode' the value of k is defined by the register bits *KLUMAFIX* and *KCHROMAFIX* while in adaptive mode k is controlled by the amount of motion found. Motion is determined by the local (low pass filtered) difference between the original and the field delayed picture. Except for settings like k = 1 (noise reduction off) or k = 0 (frozen picture) a fixed k-factor is not recommended since it results in wiping or smearing of moving objects. The dependency of the k-factor from the detected motion is defined in the k-curve which is programmed by the register settings *KSTEP0 .. KSTEP7*. Fig. 28 shows a sample k-curve. In this curve the dependency of the k-factor on the pixel difference between new and old pixel is defined. When programming the curve the different weights of the *KSTEP* values have to be observed. In fig. 28 three examples are shown: *KSTEP6* and *KSTEP7* have the weight of 8, so *KSTEP6* = 4 and *KSTEP7* = 6 gives k-values of 32 and 48 resp., while *KSTEP5* = 6 with a weight of 4 gives 24. A complete overview of kStep, k, and related weight is given in fig. 29.

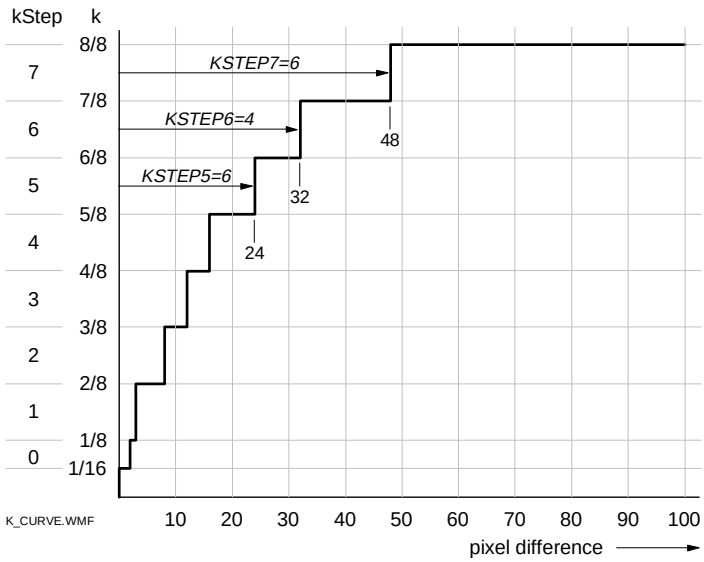


Fig. 28 Sample noise reduction k-curve

kStep..	k = ...	weight
kStep0	1/16 ... 1/8	1
kStep1	1/8 ... 2/8	1
kStep2	2/8 ... 3/8	2
kStep3	3/8 ... 4/8	2
kStep4	4/8 ... 5/8	4
kStep5	5/8 ... 6/8	4
kStep6	6/8 ... 7/8	8
kStep7	7/8 ... 8/8	8

Fig. 29 Defining a k-curve

The effective noise reduction is furthermore influenced by the gain settings *YADAPT_GAIN* and *CADAPT_GAIN* which reduce or amplify the measured pixel differences before they are used for the k-curve look-up table. Varying these register settings from 0 to 7 will give a gain setting of 1/8, 1/4, 1/2, 1, 2, 4, 8 and 12.

The calculation of k is done in both the luminance and the chrominance channel, so the amount of averaging can be defined independently for both channels. However the chrominance averaging can also be slaved to the luminance averaging by setting parameter *KLUMATOCHROMA* = 1. This for example results in an effective decrease of cross color patterns where differences from field to field are only present in the chrominance channel due to the alternating color phase.

A switchable band split filter gives the opportunity to reduce noise only in the lower half of the video spectrum, the upper band remains unchanged. This results in better picture performance without "smearing", particularly at strong settings of other parameters. *UNFILTERED* = 1 turns this filter off.

The required calculations in the recursion loop have only limited accuracy. Remaining errors thus can circulate in the loop without being further reduced. At sudden scene changes this can lead to so-called 'left over images',

faintly visible images of the previous scene. This problem is overcome by turning on the function *NOISESHAPE*. This activates an additional algorithm which eliminates the 'left over image' problem.

4.1.6 Noise estimator

Measuring noise in a video signal would preferably be done in a part without picture content like the vertical or horizontal blanking period. However, measurement here is not reliable because of possible artificial signal content, e. g. new blanking insertion at VCR playback. So in the SAA4979 noise measurement is carried out within the active video signal. Because noise is hard to detect in moving parts of a picture with a high degree of detail information, the task is to find those parts of a picture that have almost no detail (flat areas).

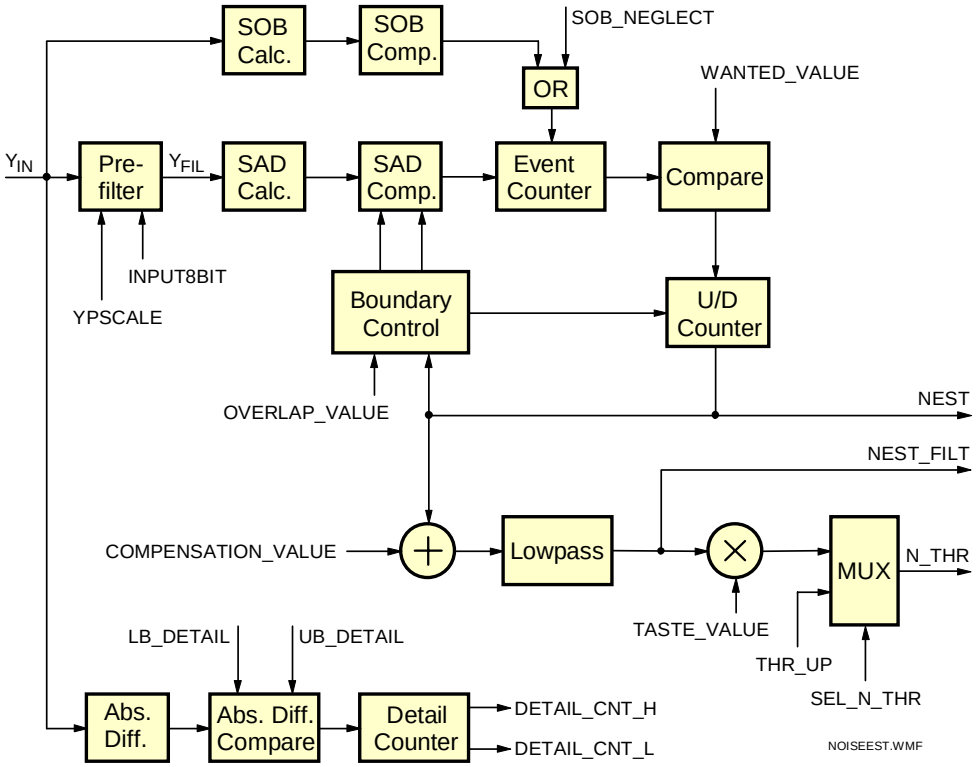


Fig. 30 Block diagram of noise estimator

A block diagram of the noise estimator is given in fig. 30. In the PREFILTER block the interesting part of the spectrum is boosted in order to increase the sensitivity of the noise measurement circuit for video sequences with low noise level. With Y_{FIL} being the output signal of the filter, parameter *YPSCALE* has the following influence:

YPSCALE = 0: scale = 1
Y*PSCALE* = 1: scale = 1/2
YPSCALE = 2: scale = 1/4
YPSCALE = 3: $Y_{FIL} = Y_{IN}$ (Filter off)

The idea of the noise estimator is to find flat areas in the picture and to determine the noise there. In order to find these areas each pixel is compared in amplitude to its neighboring pixels. In the block SAD calculation (SAD = Sum of Absolute Difference) the absolute values of the differences between the actual prefilter output Y_{FIL} and the four previous ones are summed up. These sums are then compared to a lower and upper bound. Each sum within these limits increments the event counter.

The event counter can be disabled depending on the result of the SOB (SOB = sum over block) calculation and comparison. This function permits to detect black (level below a lower clipping level *lclip*) or white areas (level above an upper clipping level *uclip*) in the picture (e. g. black bars or side panels) which may have a signal content not representative for the picture. The interval between *lclip* and *uclip* can be varied by the control parameter *CLIP_OFFSET*, see fig. 31. Parameter *SOB_NEGLECT* = 1 means that the result of the SOB comparison is not used and noise measurement is carried out in the complete video range, *SOB_NEGLECT* = 0 turns off measurement around the white and black level.

clip_offset	*real offset*	*lclip*	*uclip*
0	1	17	238
1	2	18	237
2	4	20	235
3	8	24	231

Fig. 31 Clipping levels in the SOB calculation

At the end of every field the value of the event counter is stored and the counter is reset. The stored number of events is compared to the user defined *WANTED_VALUE*. If the number of events is smaller than *WANTED_VALUE* then a 4-bit up/down counter is incremented, otherwise it is decremented. The contents of the up/down counter is the noise estimator value *NEST* and can be read by the microprocessor. It is also used as input to the boundary control block where the lower and upper limits for the SAD comparison are adjusted. In this way the control loop is closed and the sensitivity can be influenced by *WANTED_VALUE*. The lower and upper limits are furthermore adjusted by the parameter *GAIN_UPBND* which determines the difference between these two limits, see fig. 32.

GAIN_UPBND	*lobnd_del*	*upbnd*
0	0..15	1.5 * lobnd + 1
1..6	0..15	lobnd * GAIN_UPBND + 1
7	0..3	1.5 * lobnd + 1
7	4..15	lobnd * (0.5 * lobnd)

lobnd_del . . . lower boundary of previous field
upbnd . . . upper boundary
GAIN_UPBND . . . control input for calculating upbnd

Fig. 32 Calculation of the interval upper boundary *upbnd*

It is difficult to avoid that scenes with a large amount of detail result in a higher noise estimate compared with a scene of less detail but the same amount of noise. Therefore a function to measure the detail is incorporated. The difference between every two adjacent pixels is taken and compared to *LB_DETAIL* and *UPB_DETAIL*. The number of times in a picture where this difference falls in between these boundaries is counted and can be read by the microprocessor in a two byte value: *DETAIL_CNT_H* and *DETAIL_CNT_L*. The microprocessor evaluates these data and calculates a correction factor *COMPENSATION_VALUE* for the noise estimation.

The noise estimate *NEST* is also available in a lowpass filtered version: *NEST_FILT*. After a possible compensation offset a lowpass filter generates a moving average over the last 16 *NEST* values. The filter is recursive and generates a 13 bit value which is scaled back to 8 bits for output as *NEST_FILT*.

4.2 3.5 MBit field memory

The SAA4979 has a built-in scan conversion memory. The memory is similar to the SAA4956. The main difference is its data width of 16 bits instead of 12, so now video data in 4:2:2 format can be processed. The field memory is capable to store for example up to 307 video lines of 720 pixels in 4:2:2 format. It has one write interface (controller and registers) to store $1f_H$ data and two read interfaces, one to read field delayed $1f_H$ data for the

noise reduction function and the other to read $2f_H$ data for the following data processing, see noise reduction block diagram in fig. 27. Since two asynchronous clock domains are involved ($SWCK_{int}$ as $1f_H$ clock and $SRCK_{int}$ as $2f_H$ clock) the read and write access to the memory array is controlled asynchronously by the memory arbitration logic triggered via request and acknowledge pulses.

The memory has internal address generators for writing and reading. The write address pointer is reset by the $RSTW_{int}$ pulse which is derived from the 50 Hz vertical sync pulse V656_1, the read address pointer is reset by the $RSTR_{int}$ pulse which occurs in the vicinity of the vertical 100 Hz sync pulse VD. Whenever WE_{int} is active, data is written to the memory, and whenever RE_{int} is active, data is read from the memory. WE_{int} and RE_{int} are generated by the memory controller within the SAA4979. Fig. 33 shows a block diagram of the scan conversion memory. ($RSTW_{int}$, V656_1, WE_{int} and RE_{int} are chip internal signals).

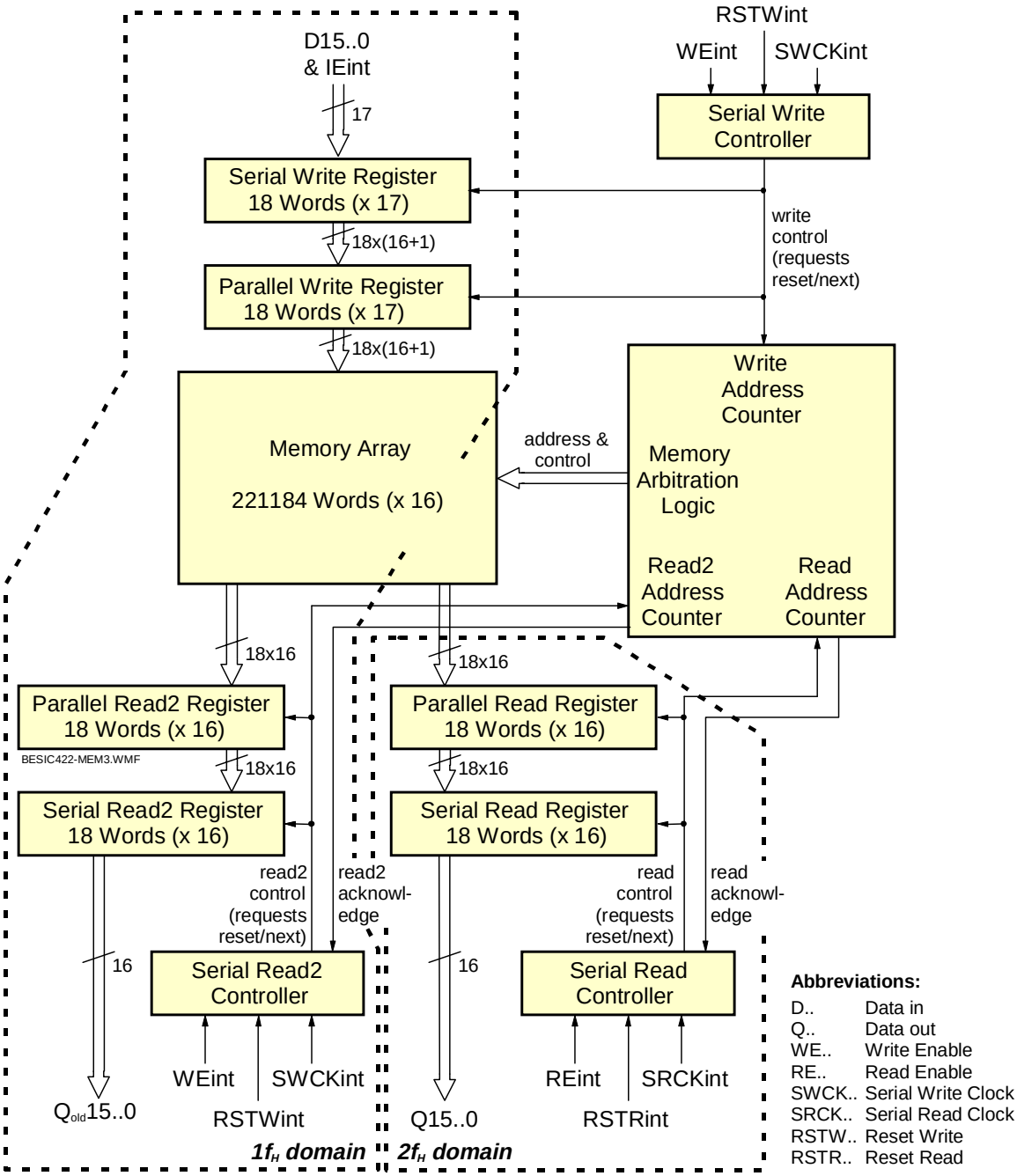


Fig. 33 Block diagram of the scan conversion memory

The $RSTW_{int}$ pulse can be shifted (delayed) with respect to the V656_1 pulse, this gives a vertical shift of the picture on the screen. This is applied together with vertical zooming for example when the center part of the picture (without the black bars) is to fill the screen.

4.3 Digital processing at $2f_H$ level

4.3.1 Sample rate conversion

The fixed sample rate conversion (FRSC) block is used to obtain 848 active pixels per line out of the original 720 pixels according to the relation of the two sampling frequencies (32 MHz and 27 MHz). The interpolation for phase positions between the original samples is achieved with a variable phase delay filter with 10 taps for the luminance signal and 6 taps for the chrominance signals.

The conversion to a higher sample frequency of 32 MHz is done to improve the motion estimation performance in combination with external feature ICs, which can process up to 848 pixels per line at a 32 MHz clock. Bypassing this function keeps the original 720 pixels per line (control input: *BYPASS_FSRC*).

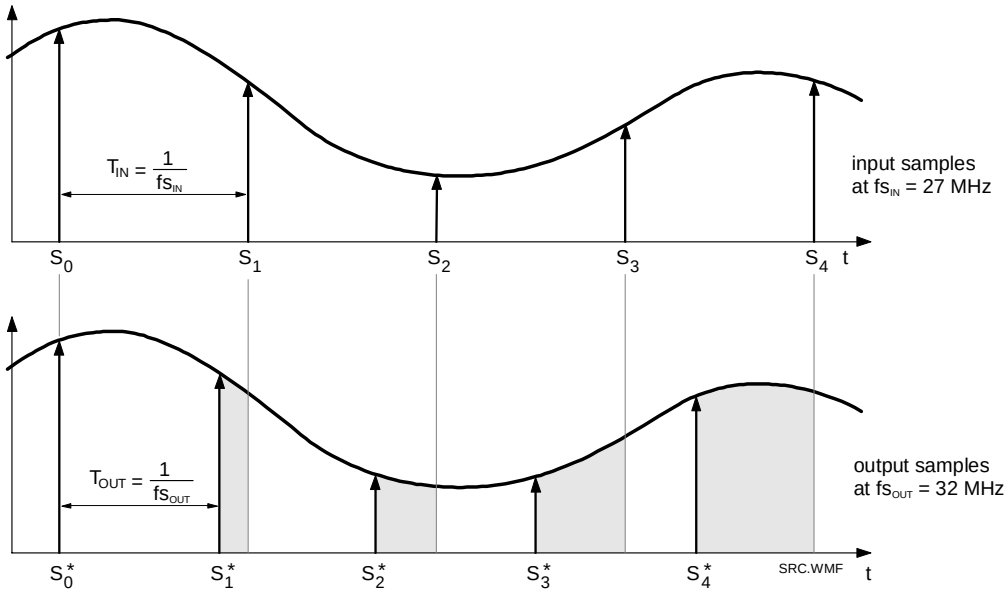


Fig. 34 Sample rate conversion by interpolation

4.3.2 Expansion Port

For particularly economic 100 Hz solutions the SAA4979 can be used as a stand-alone device offering one-chip 100 Hz. There is however also an expansion port in order to connect further picture enhancement ICs like the SAA4991 (MELZONIC) or SAA4992 (FALCONIC). The port can be configured for 4:1:1 data format (if the SAA4991 is used) or 4:2:2 format (if the SAA4992 is used).

Fig. 35 shows a block diagram of the output part of the expansion port. Only the chrominance signal *UV* is processed, the luminance signal *Y* is unchanged.

If needed, the incoming chroma signal *UV_IN* can be inverted in the block UV_INVERTER, in this case the control signal *MID_UV_INV* must be active. Of course, the output signal is limited to +127 to prevent an overflow from inverting the minimum signal value of -128.

In order to provide data in 4:1:1 format, the chroma signal is first downsampled in block DOWN_422_411. Here the bandwidth is reduced by a factor of 2. Then it is formatted to 4:1:1 by the FORMAT block. Setting the control

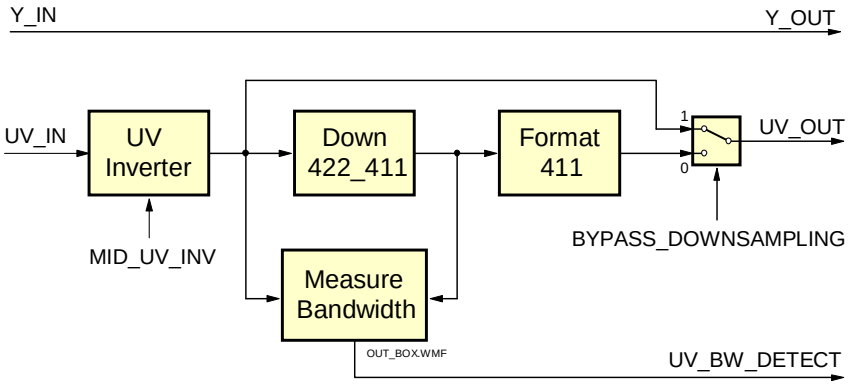


Fig. 35 Block diagram of the output part of the expansion port

signal *BYPASS_DOWNSAMPLING* to 0 will put the formatted signal through to the output, setting the control signal to 1 will leave the format unchanged and 4:2:2 data is output.

In the block MEASURE_BANDWIDTH the 4:2:2 chroma data from the output of the downsampling filter is compared to the data at the input of the filter. If considerable differences are found this is an indication for 4:2:2 chroma bandwidth. The information can be read by the microcontroller and can help in automating chroma settings, e. g. for CTI (color transient improvement). Bandwidth detection is done in a programmable window defined by the control signals *BW_HSTART*, *BW_HSTOP*, *BW_VSTART* and *BW_VSTOP*.

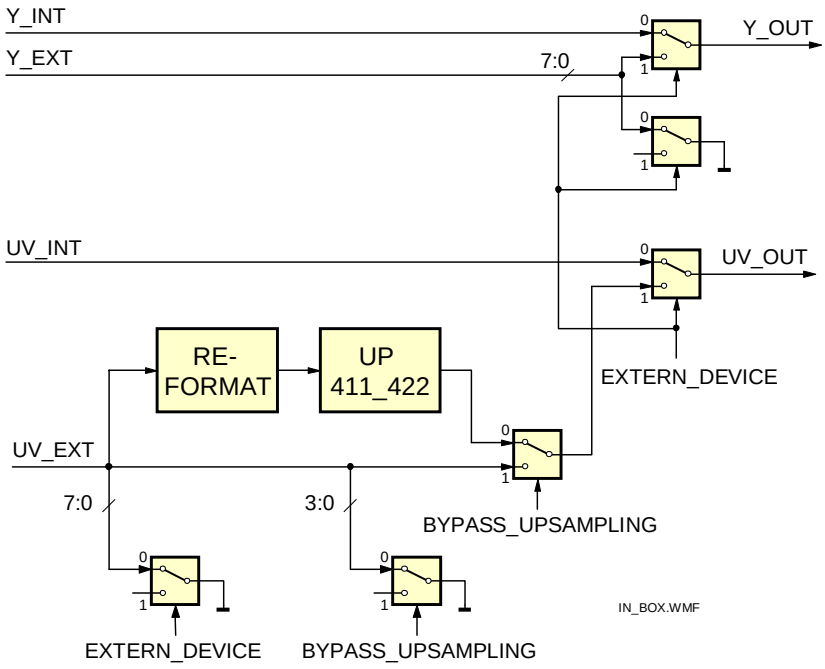


Fig. 36 Block diagram of the input part of the expansion port

Fig. 36 shows a block diagram of the input part of the expansion port. Only the chrominance signal *UV* is processed, the luminance signal *Y* is unchanged. The control signal *EXTERN_DEVICE* = 1 selects data from the expansion port to be used in the back end part of the IC. If *EXTERN_DEVICE* = 0 the internal signals Y_INT and

UV_INT are put through and the external signals are grounded. Since the internally used signal format is always 4:2:2, in case of external signal a reformatting and upsampling may be necessary if the format is 4:1:1. This is selected by the control signal *BYPASS_UPSAMPAMPLING* = 0. According to the 4:1:1 format in this case only four bits of chrominance are input, so the unused four input bits are put to ground. *BYPASS_UPSAMPAMPLING* = 1 is used for external 4:2:2 data.

4.3.3 Horizontal Zoom, Panorama

This block essentially consists of a variable delay line of which the delay can be dynamically controlled with sub-pixel accuracy. The several modes of operation will be explained with reference to the relative input sample rate Sr, being the instantaneous ratio between the sample frequencies at input and output:

$$Sr(x) = \frac{\text{input sample frequency}}{\text{output sample frequency}} = \frac{\text{output sample period}}{\text{input sample period}}$$

- Horizontal shift

  Horizontal shift is done by simply delaying or advancing the incoming lines. The range for this shift is from -1/2 line to +1/2 line (plus the nominal delay of 1/2 line). It is controlled by the parameter *H_SHIFT* (16 bit).

- Linear compression and expansion

  With a zero order variation of the delay a linear compress or expand function is obtained. The range for the compression factor is 0 to 2, meaning infinite zoom to a compression factor of 2. The amount of compression or expansion is determined by the parameter *C0* (zero order control). In fig. 37 a the shaded area is equal to the total mount of input samples converted to output samples.

- Nonlinear compression (panorama) and nonlinear expansion (amaronap)

  With a second order variation of the delay a parabolic compression or expansion is obtained. This means that the lines are geometrically expanded at the sides and slightly compressed at the centre (see fig. 37b). This mode is especially useful for display of 4:3 pictures with full width on a 16:9 screen, whereby the geometry in the centre is more or less correct (see fig. 38).

  The required modulation of the sampling period is parabolic. The amplitude of the parabolic modulation is determined by the parameter *C2* (second order control) whereas the compression factor at the centre of the line is controlled by *C0*.

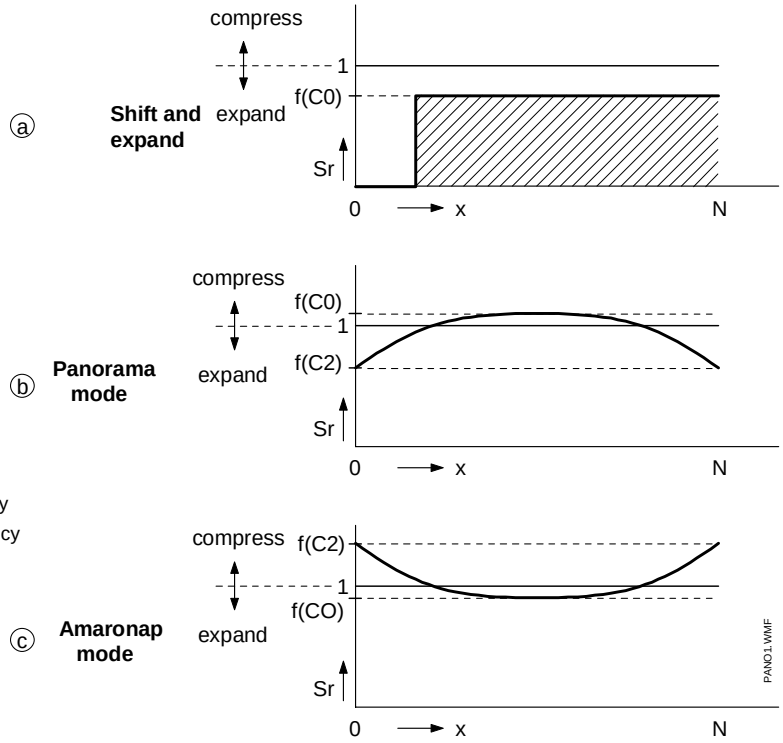
  The amaronap mode (see fig. 37 c) is the inverse of the panorama mode. It can be used for full width display of 16:9 pictures on a 4:3 screen.

Fig. 38 show the possible effects on the screen: when a 4:3 picture is displayed on a wide screen it is distorted (expanded horizontally). Static compression is required in order to restore the correct aspect ratio. In this case only part of display area is used and there will be side bars on the screen (usually black). A possible solution to utilize the complete display area is panoramic zoom. Here the center section (assumed to contain the most important parts of the picture) nearly has its correct aspect ratio while towards the sides the expansion is gradually increased. Control parameter *C0* influences the aspect ratio in the center and parameter *C2* permits to adjust the amount of expansion increase towards the sides.

The registers to set *H_SHIFT*, *C0* and *C2* are only accessible by the internal microcontroller. Instead, the firmware offers different predefined settings for horizontal zoom and compression, see bits 3..5 of register 01_H (Field_Control_2).

4.3.4 Digital Color Transient Improvement (DCTI)

The Digital Color Transient Improvement (DCTI) is originally intended for U and V signals originating from a 4:1:1 source but 4:2:2 will also benefit from this circuit. The basic principle is to detect horizontal transients and improve their steepness without generating overshoots. This principle is depicted in fig. 39. U and V data always



$$Sr(x) = \frac{fs_{input}}{fs_{output}}$$

fs_{input} ... input sample frequency

fs_{output} ... output sample frequency

Fig. 37 Principle of panoramic zoom

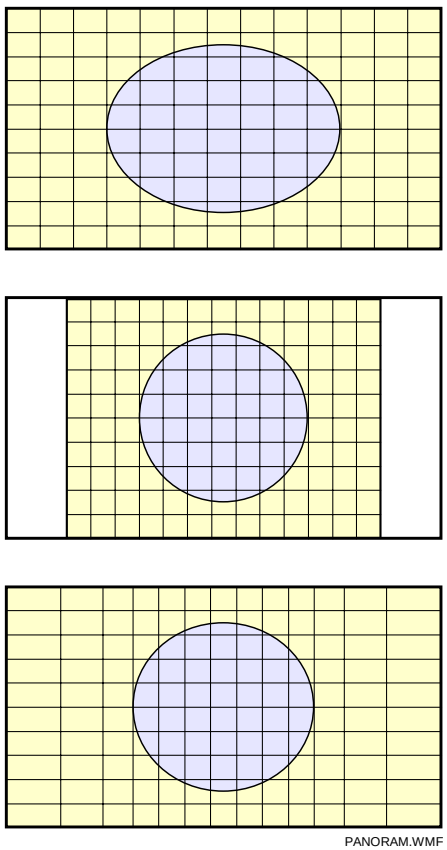


Fig. 38 Nonlinear compression/expansion in panorama mode

enter the block in 4:2:2 format but regarding their bandwidth they may stem from a 4:1:1 source. During the process upsampling to the 4:4:4 format occurs.

The idea is to vary the data path delay on the basis of a function of the second derivative of the U and V signal. Positive and negative transients are treated alike, the output of the first differentiator therefore is taken as absolute value. The signal is differentiated again and the output used to control the momentary data path delay. The effect at an edge is that during the first half the data path delay is higher than nominal and in the second half it is lower than nominal. This will make the edge much steeper. The control signal that varies the delay is amplified by a user defined gain setting (*DCTI_GAIN*). Increasing this parameter results in a steeper transient.

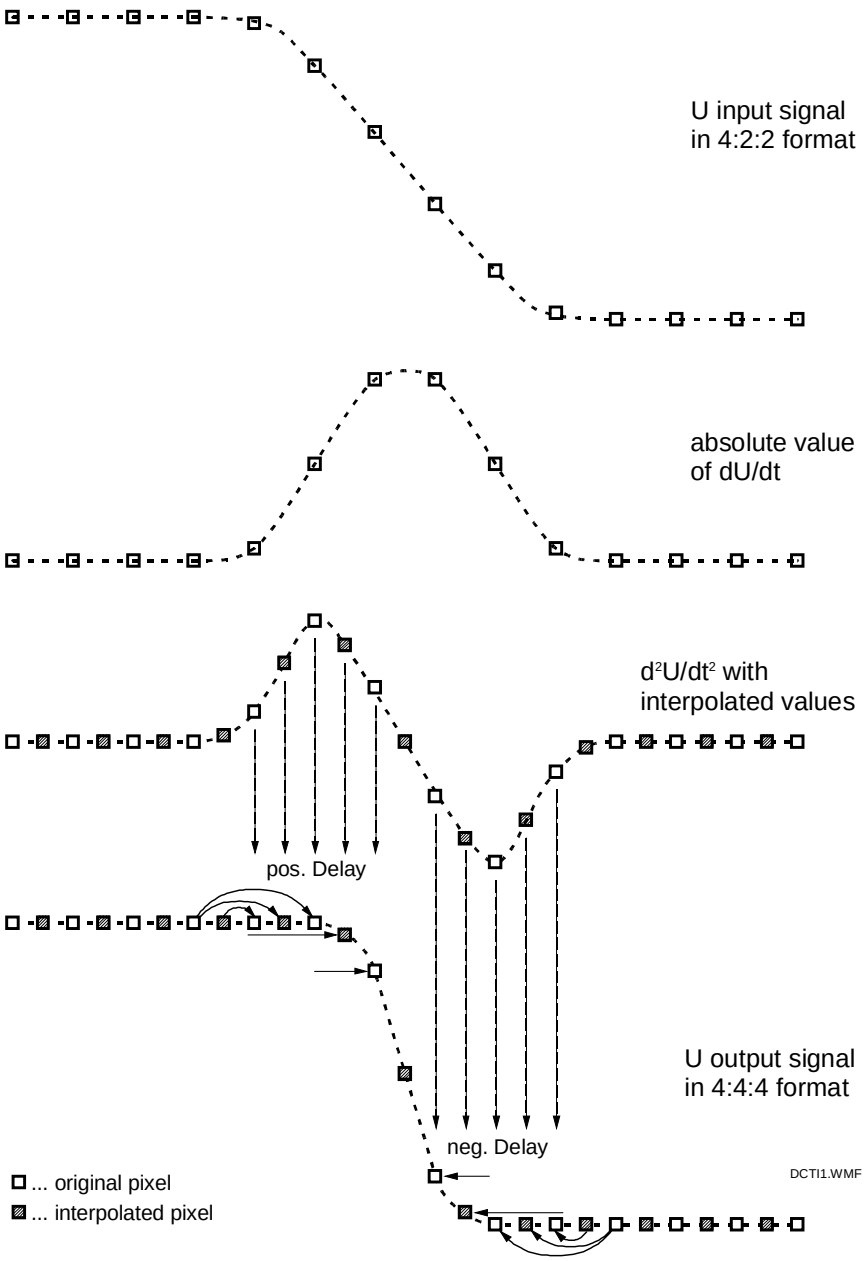


Fig. 39 DCTI basic operating principle

There are two differentiating filters in order to obtain the second derivative. The first differentiating filter calculates the first derivative of the U and V signals. The filter offers two transfer curves which can be selected by the

parameter *DCTI_DDX_SEL*. The transfer curves are given in fig. 40. A standard quality filter with property [-1 0 0 1] is selected with *DCTI_DDX_SEL* = 0, a high-quality filter with property [-1 -2 -1 1 2 1] is selected with *DCTI_DDX_SEL* = 1.

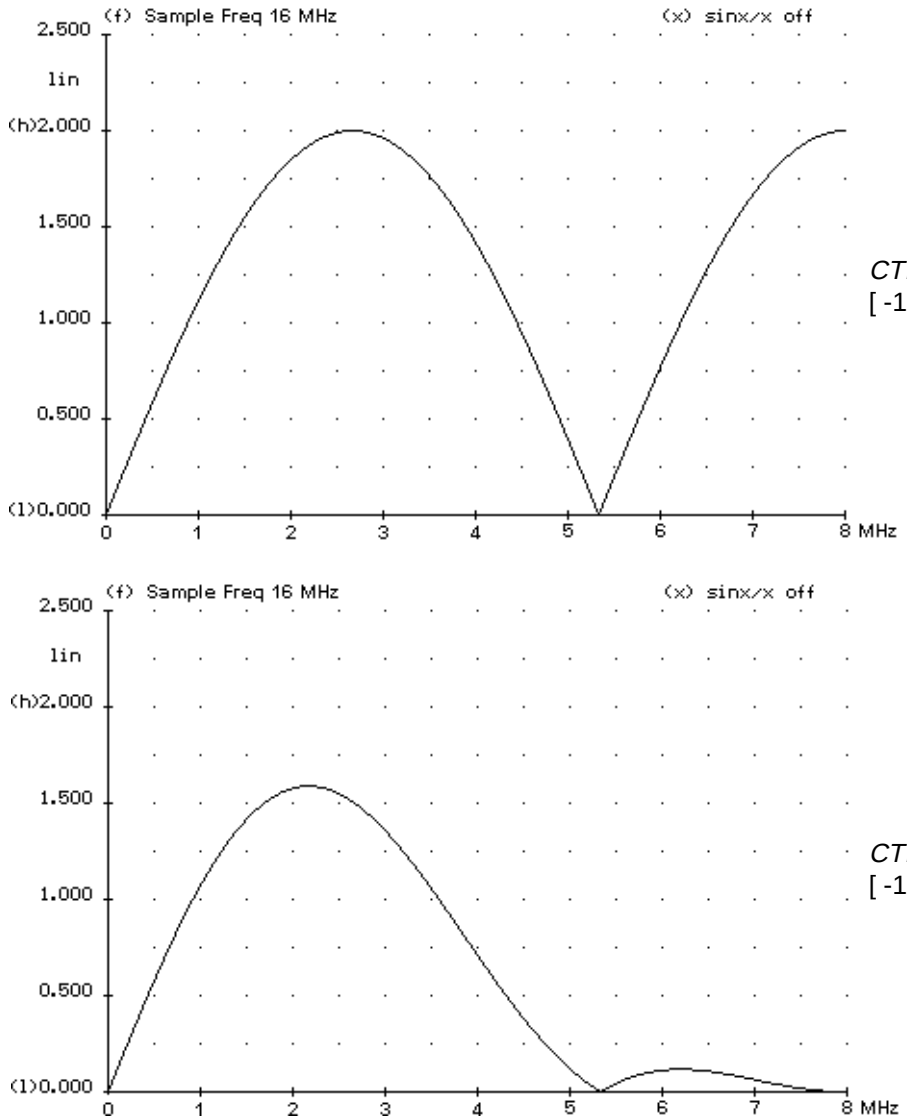


Fig. 40 Transfer curves of the first differentiating filter

The second differentiator calculates the second derivative of the U and V signals. It runs at 32 MHz sample clock and generates interpolated values. The original U and V signals also are upsampled to 32 MHz, so the output of the DCTI circuit has a resolution equal to that of the Y signal. The output of this second differentiator is, except for gain and clipping, the drive signal for DCTI. Because the input is the absolute value of the first differentiator the output has the right polarity: negative for the left side of the ramp and positive for the right side of the ramp.

The DCTI function can be controlled mainly by adjusting the parameters *DCTI_GAIN* and *DCTI_LIMIT*. *DCTI_GAIN* influences the resulting steepness of the output signal. A selection can be made from a gain of 0 to a gain of 7/8 in steps of 1/8. When setting the gain parameter to 0 then DCTI is switched off and the POSTFILTER should be activated to correct the upsampling. Modification of this parameter is depicted in fig. 41 using a maximum amplitude color transient as input signal.

DCTI_LIMIT affects the maximum amount of data path delay. User definable values for this parameter are 0, ±4, ±8 and ±12. Modification of this parameter is depicted in fig. 42 using a maximum amplitude color transient as input signal. Both *DCTI_GAIN* and *DCTI_LIMIT* must be greater than zero for DCTI to be active.

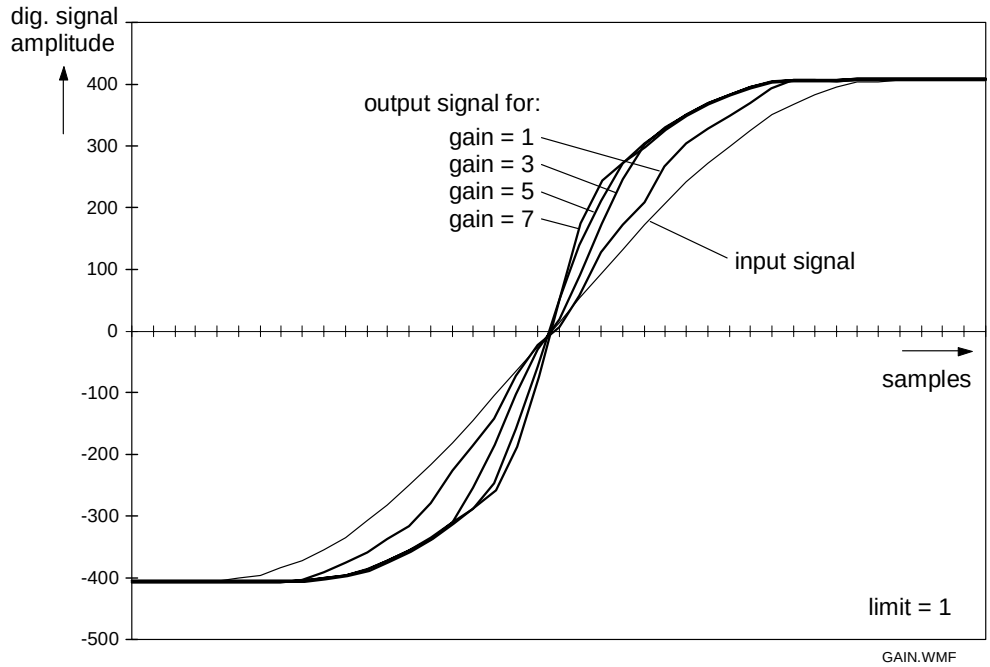


Fig. 41 DCTI with variation of gain for a limit setting of 1

An artifact of this processing becomes apparent when two edges are close together in the video signal. During the second half of the first edge a delay is chosen that will collect video data from where the second edge is already active. The same is valid for the second edge. The result of this processing on a video pulse, which is looking like a hill, is that of a hill with one or two bumps on it. To prevent this from happening, the positions where the first derivatives in U and V change sign, are marked and used to limit the range of the relative delay. This function is called 'over-the-hill protection'. It can be turned on and off by the parameter *DCTI_PROTECTION*. Fig. 44 and 45 show the effect of the DCTI function with and without 'over the hill protection' when applied to a hill-shaped video pulse. In order to detect a hill the second derivative of the input function is checked for a sign change (zero crossing), see fig. 43. When a hill is detected the distance to that hill for each directly surrounding pixel is calculated. The drive signal will be dynamically limited to this distance for each pixel. The result is that DCTI is prevented from 'looking over the hill'. For hill detection a threshold can be set by the 4-bit parameter *DCTI_THRESHOLD*.

The 'hill protection' function still produces artefacts for signal transitions where the first derivative does not change sign, i. e. two (or more) positive (or negative) steps following each other. Signals of this kind are handled properly if 'superhill-protection' is turned on by parameter *DCTI_SUPERHILL* = 1. The behavior of DCTI with active and inactive 'superhill protection' is shown in fig. 46 and 47. Slight overshooting occurs if the postfilter is turned on.

The postfilter is used to correct upsampling in case DCTI is not activated (*DCTI_GAIN* = 0 and/or *DCTI_LIMIT* = 0). In this case upsampling uses only linear interpolation and the output signal shape can be improved by turning on the postfilter. The transfer characteristic is given in the upper curve of fig. 48. The lower curve gives the corrected upsampling characteristic with the postfilter turned on. The postfilter can be turned on by the parameter *DCTI_FILTERON* = 1.

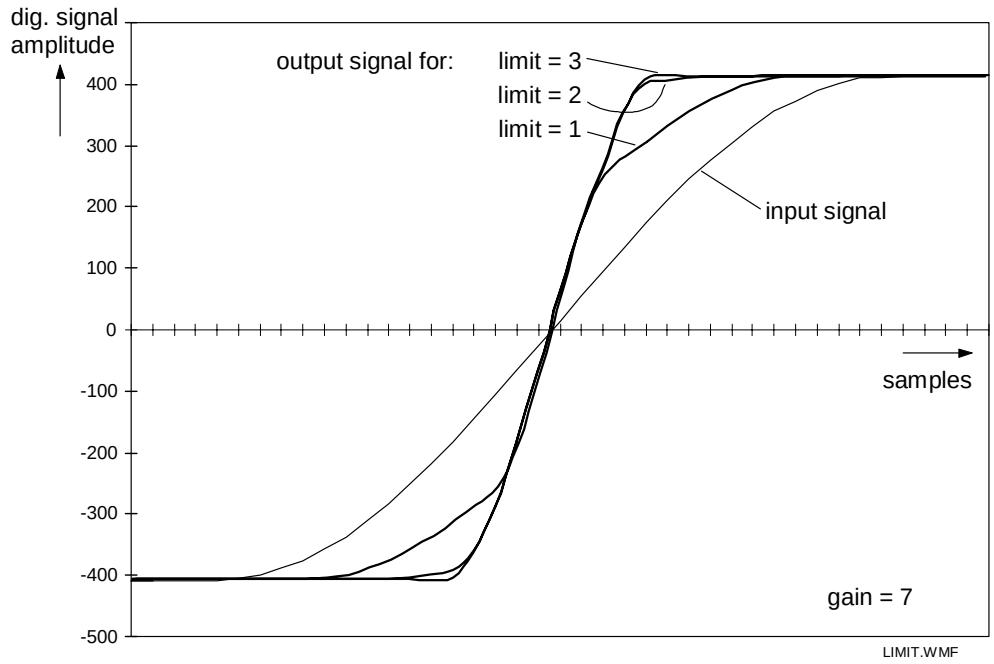


Fig. 42 DCTI with variation of limit for a gain setting of 7

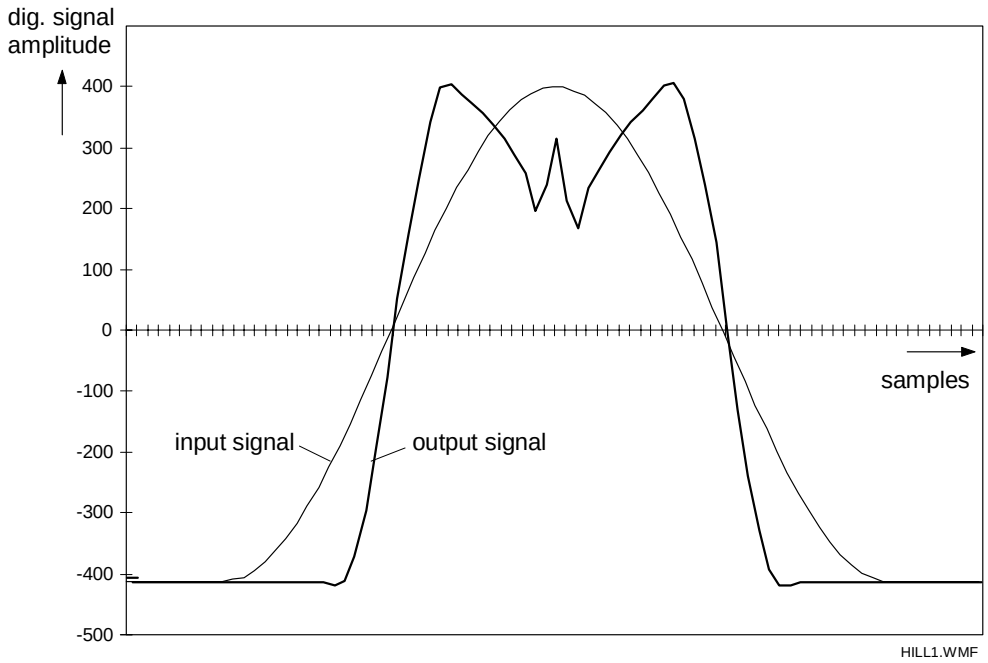


Fig. 44 DCTI without ´over-the-hill protection´

The DCTI function can further be controlled by the parameter *DCTI_SEPARATE* in regard to whether both signals U and V are processed together, or each one separately. In case of *DCTI_SEPARATE* = 0 (off) a steep transition in either signal is sufficient to activate the data path delay variation. This setting is based on the fact

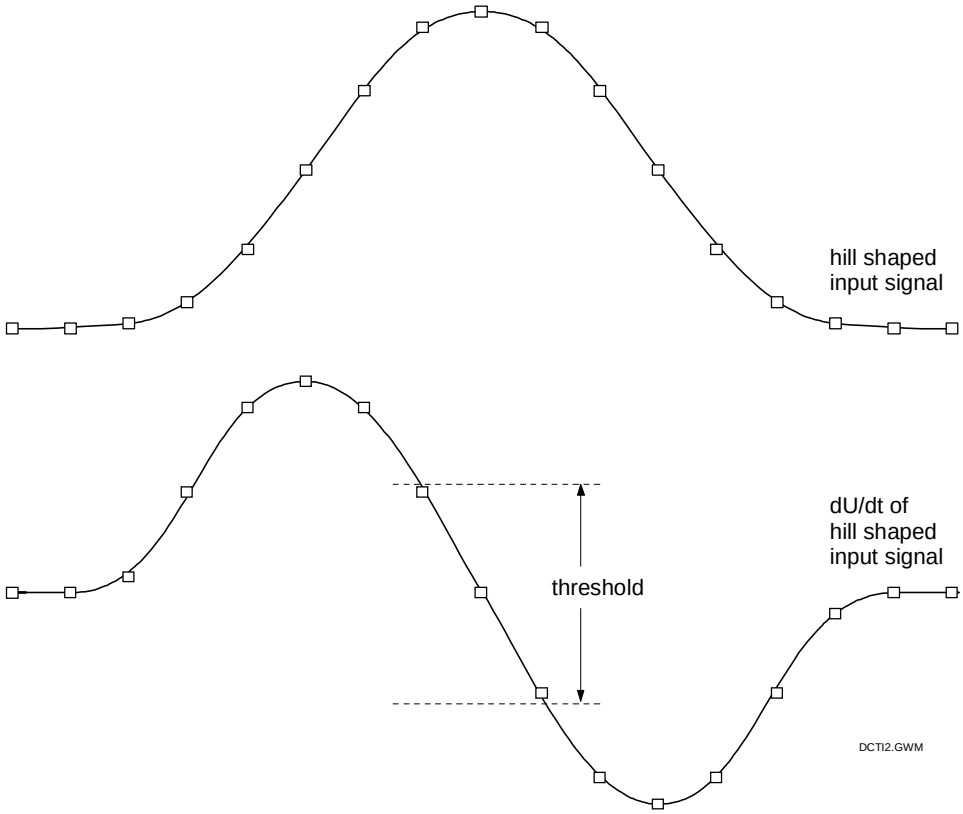


Fig. 43 Principle of hill detection

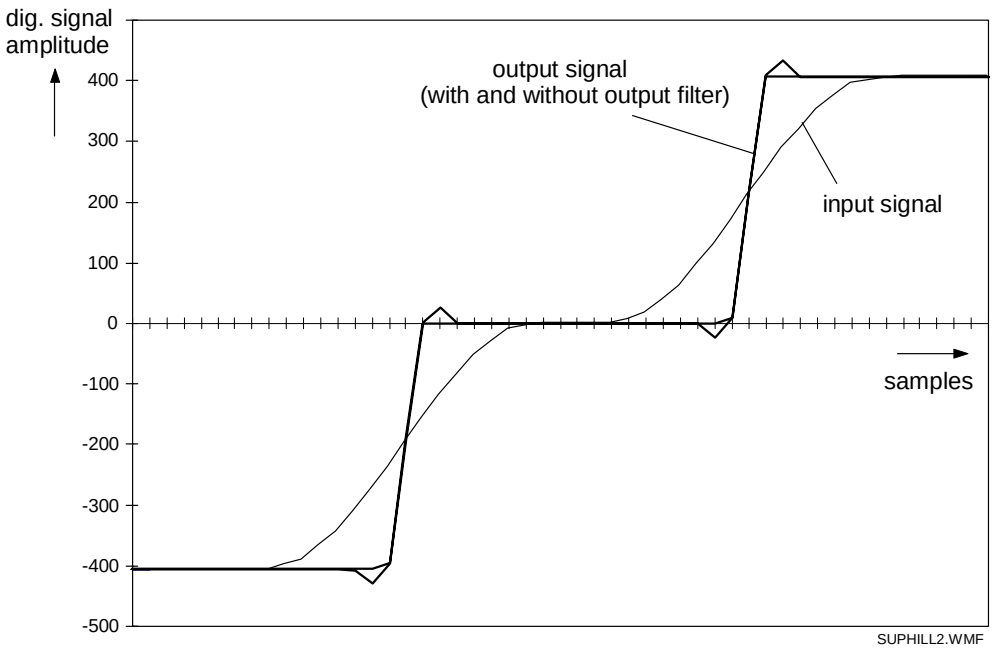


Fig. 47 DCTI with superhill-protection on

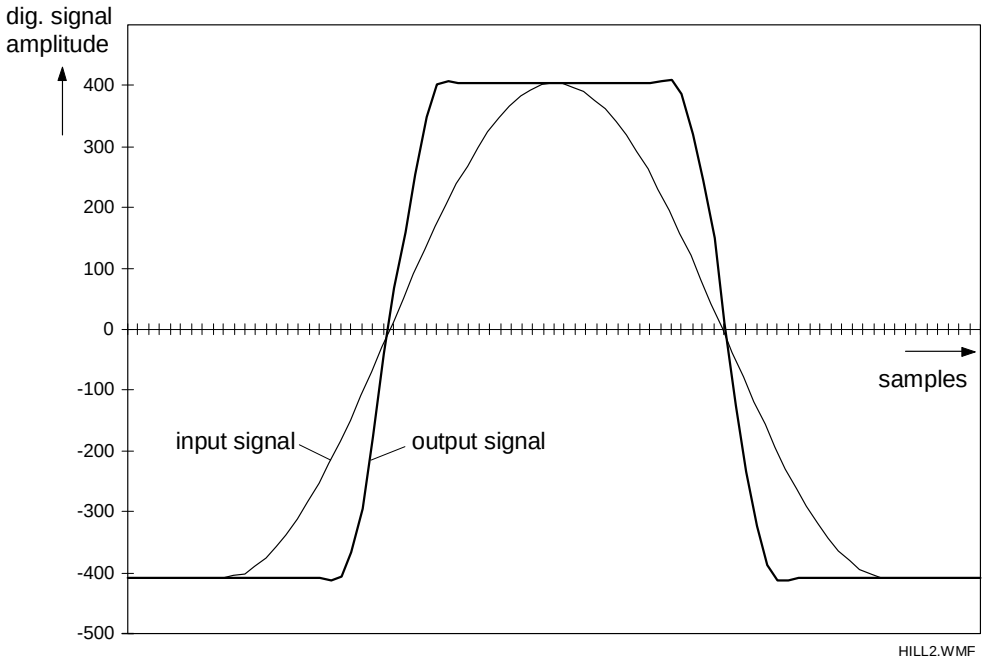


Fig. 45 DCTI with over-the-hill-protection

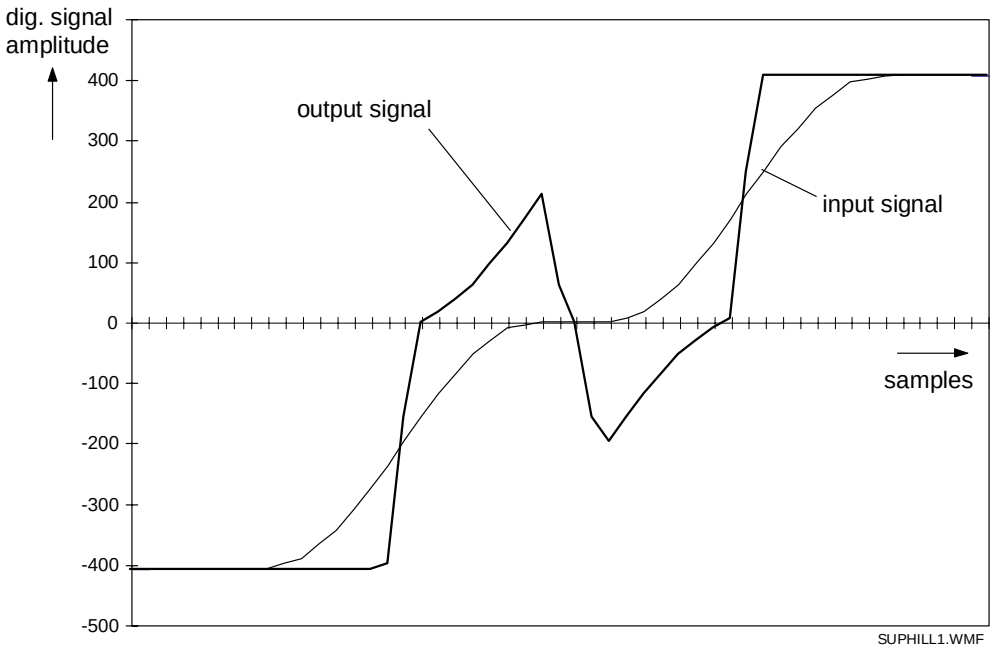


Fig. 46 DCTI with superhill-protection off

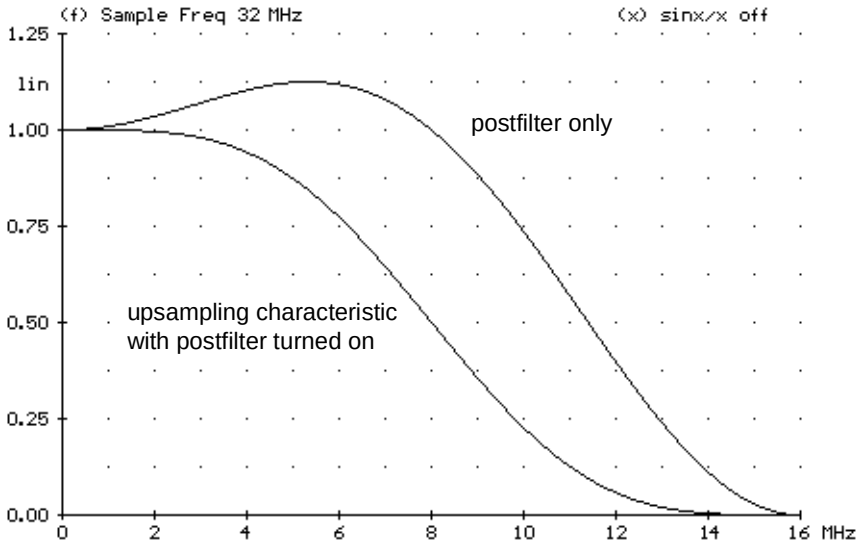


Fig. 48 Transfer curve of postfilter

that most color transients involve both signals U and V. And if one of the signals stays constant, a data path variation would do no harm.

In case of *DCTI_SEPARATE* = 1 (on) each signal is processed separately. This setting is favorable if the transitions in both signals do not occur at the same time. Common processing then would give false colors which can be annoying. An example for processing such signals is given in fig. 49 and 50.

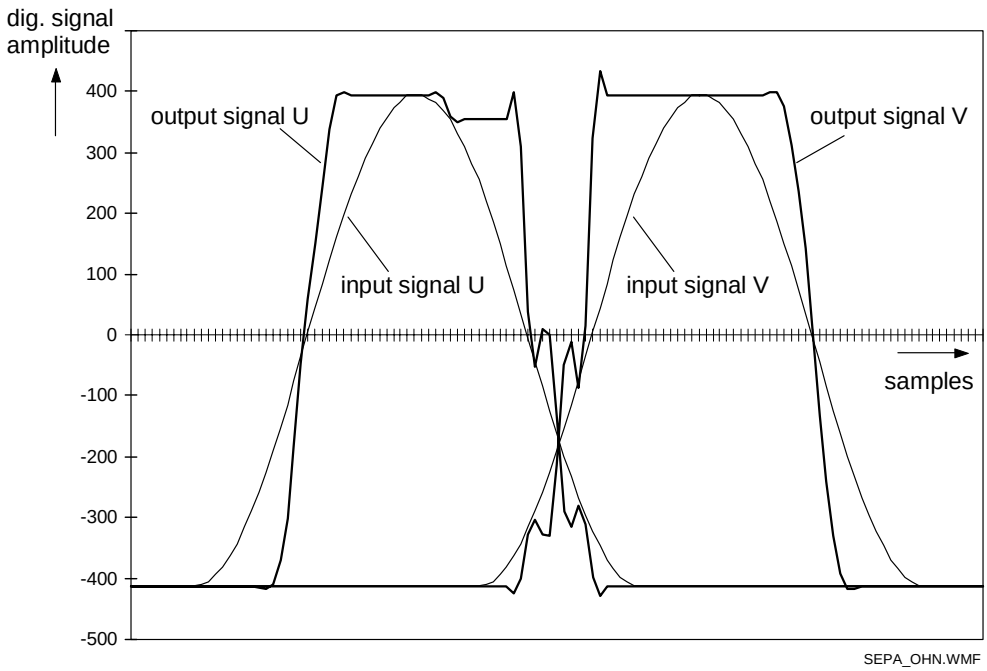


Fig. 49 DCTI with common processing of both signals (*CTI_SEPARATE* = 0)

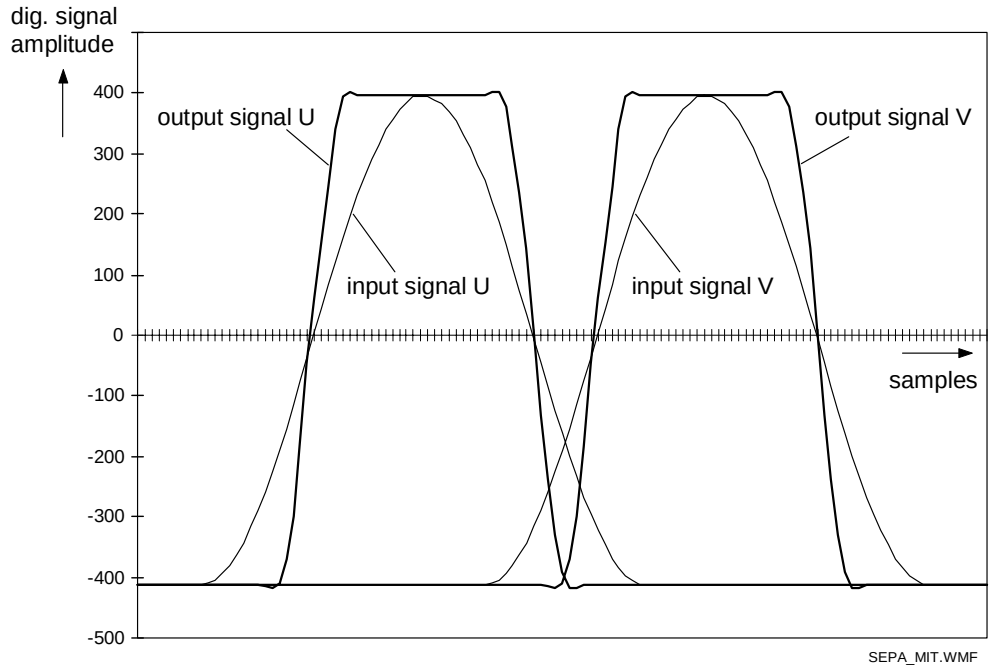


Fig. 50 DCTI with separate processing of both signals (separate = 1)

4.3.5 Y horizontal smart peaking

The luminance signal Y is processed by the peaking circuit in order to boost the higher frequency ranges. A block diagram is shown in fig. 51. The circuit uses a combination of two band pass filters and a high pass filter. The first band pass filter has the coefficients [-1 0 2 0 -1] and gives a maximum throughput at $f/f_c = 0.25$ (4 MHz)[2]. The second band pass filter is a convolution of the two filters [-1 0 0 2 0 0 -1] and [1 2 1] giving a peak at approx. $f/f_c = 0.15$ (2.4 MHz). The high pass filter is made with [-1 2 -1] coefficients with a maximum throughput at $f/f_c = 0.5$ (8 MHz). The summed output of the filters is processed by a coring circuit and then added to the original luminance signal.

The influence of each of the filters can be adjusted in eight steps from 0 to 8/16 (7/16 omitted). In fig. 52 to 54 the frequency response of each filter is given for different values of *PK_TAU* (band pass 1), *PK_ALPHA* (band pass 2) and *PK_BETA* (high pass). Fig. 55 gives an example of two transfer curves having different center frequencies.

The peaking filter will boost higher frequency signals regardless of their amplitude. For structured small signals this will lead to unwanted coring (additional noise). In order to prevent this the block WIDE CORING is added. Below a defined amplitude threshold it suppresses any gain, so the original luminance signal is not influenced. If the signal becomes larger then only the portion which exceeds the threshold can pass the coring stage. The coring threshold level can be defined by the parameter *PK_CORTHR*. This 4 bit parameter allows 16 settings from 0..120 in steps of 8. The value of 0..120 has to be seen in relation to a signal amplitude of ±1023, so the maximum setting equals roughly one eighth of the signal amplitude. The transfer curve is depicted in fig. 56.

The peaking function can be dynamically controlled in order to provide less gain on large details and edges. For this purpose the filtered luminance signal is lowpass filtered so the high pass energy is stretched to neighboring pixels in order to decrease decision noise in the attenuator. The output of the low pass filter is controlled by the parameter *PK_DELTA* which can have the values 0, 1/4, 1/2 and 1. For *PK_DELTA* = 1 attenuation is fully active, for *PK_DELTA* = 1/2 and 1/4 it is reduced and for *PK_DELTA* = 0 it is turned off. The behavior of the atten-

2. f_c ... clock frequency (16 MHz)

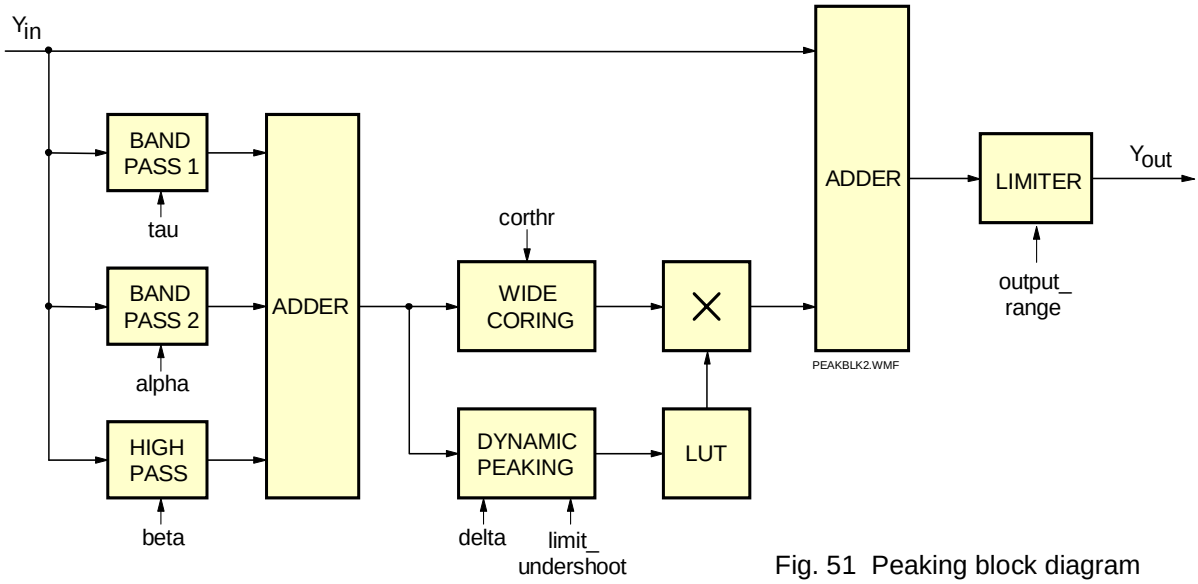


Fig. 51 Peaking block diagram

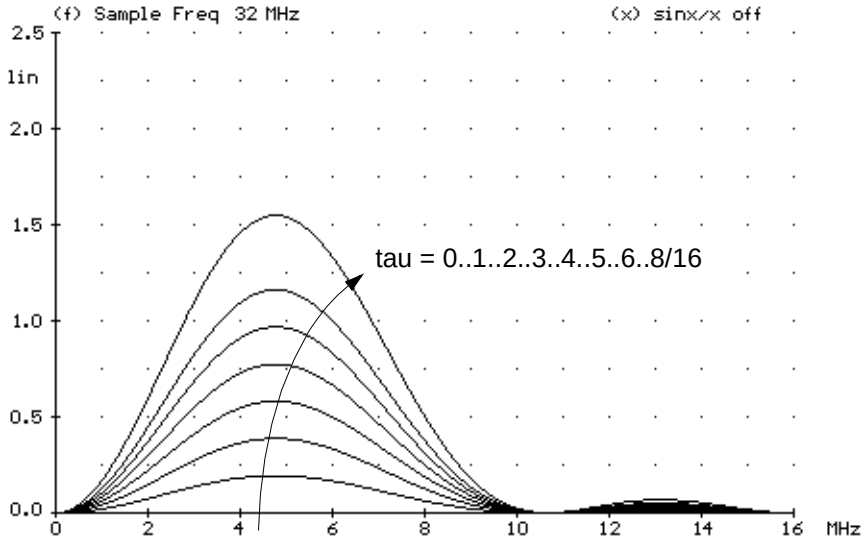


Fig. 52 Frequency response of the peaking band pass filter 1

uator is shown in fig. 57. The value of 128 is equal to no attenuation. Maximum attenuation is achieved at value 24.

In the following block NEGGAIN the negative going edges can be controlled separately. The parameter *PK_NEGGAIN* can have the values 0, 1/4, 1/2 and 1. For *PK_NEGGAIN* = 1 attenuation is fully active (same attenuation for positive and negative going edges), while for *PK_NEGGAIN* = 1/2 and 1/4 it is reduced and for *PK_NEGGAIN* = 0 it is turned off. *PK_NEGGAIN* factors of less than 1 mean that in the output signal undershoots are larger than the overshoots.

The block LIMITER limits the output signal Y to a range of 10 bits. There are two modes: *OUTPUT_RANGE* = 0 generates a nominal 9 bit signal in the 10 bit range, thus using only half of the possible output range for the nominal video content, but leaving ample room for over- and undershoots generated by the peaking circuit. Black level is at 288 and white level at 767. *OUTPUT_RANGE* = 1 makes use of 10 bits for the nominal signal, black level is at 64 and white level at 1023. Any over- or undershoots will be clipped. Fig. 58 depicts the situation.

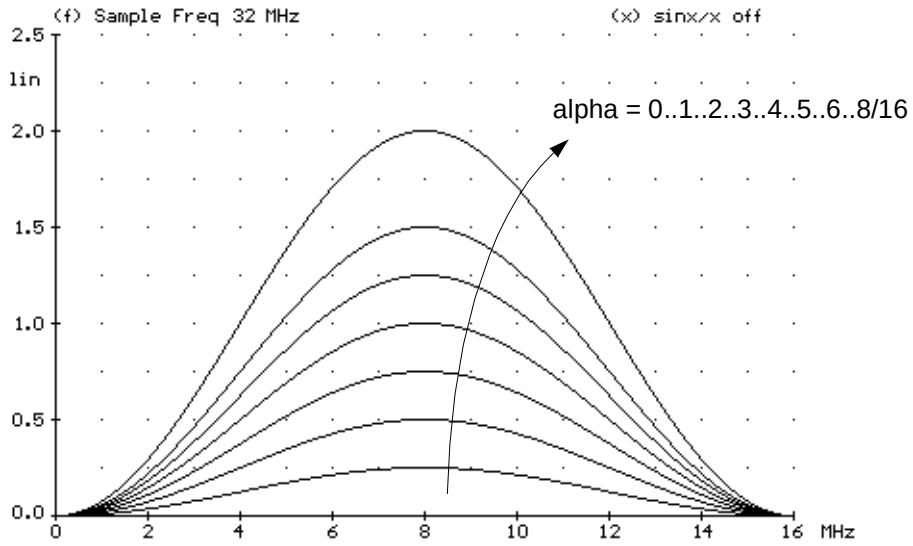


Fig. 53 Frequency response of the peaking band pass filter 2

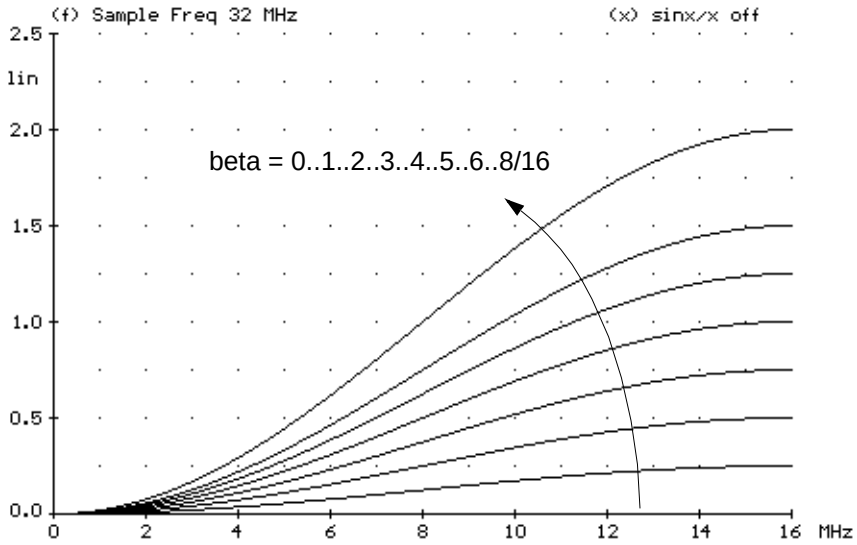


Fig. 54 Frequency response of the peaking high pass filter

The peaking steepness measurement circuit gives information about the maximum steepness of slopes in the actual field. The measurement is only active within the measurement window which is defined by the parameters *STEEPNESS_VSTART* and *STEEPNESS_VSTOP* in steps of four lines as well as *STEEPNESS_HSTART* and *STEEPNESS_HSTOP* in steps of four pixels. The output of the bandpass filter 2 is taken and the maximum value that occurred within the window is stored and output as *STEEPNESS_MAX*.

4.3.6 Non-linear phase filter

The nonlinear phase filter (NLP-Filter) is designed to compensate for nonlinear filtering and bandwidth loss at the output of the IC as well as for sin x/x compensation. The filter can be adjusted by two parameters: λ defines the highpass amplitude, and μ determines the overshoot behavior. Settings are provided for λ = 0, 1/8, 2/8 and

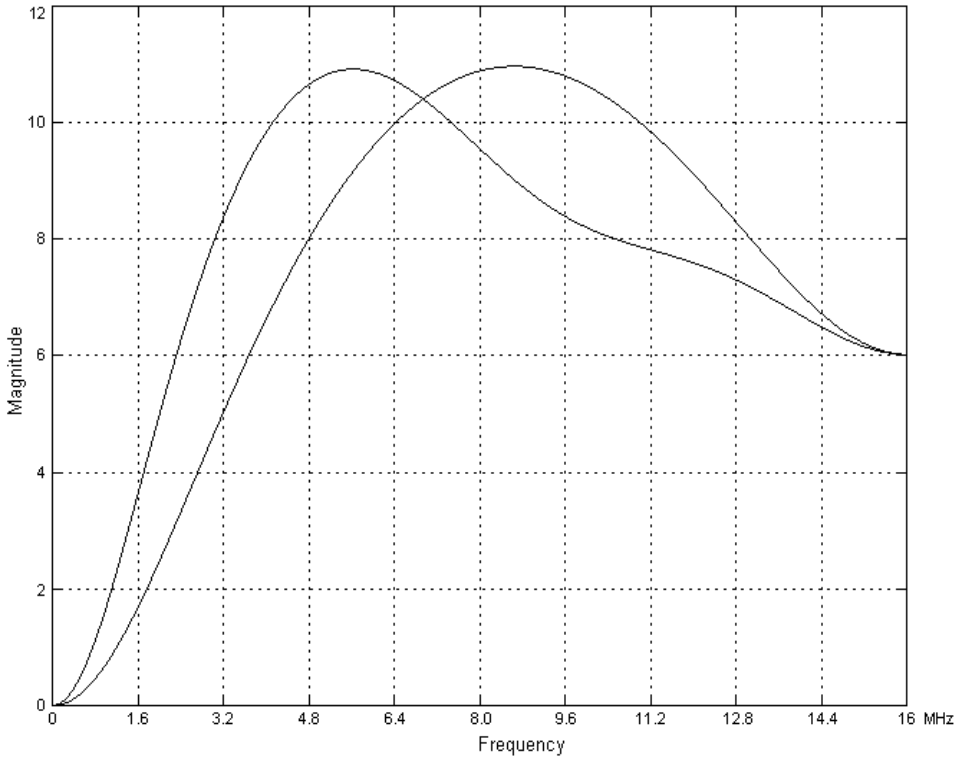


Fig. 55 Variation of peaking center frequency

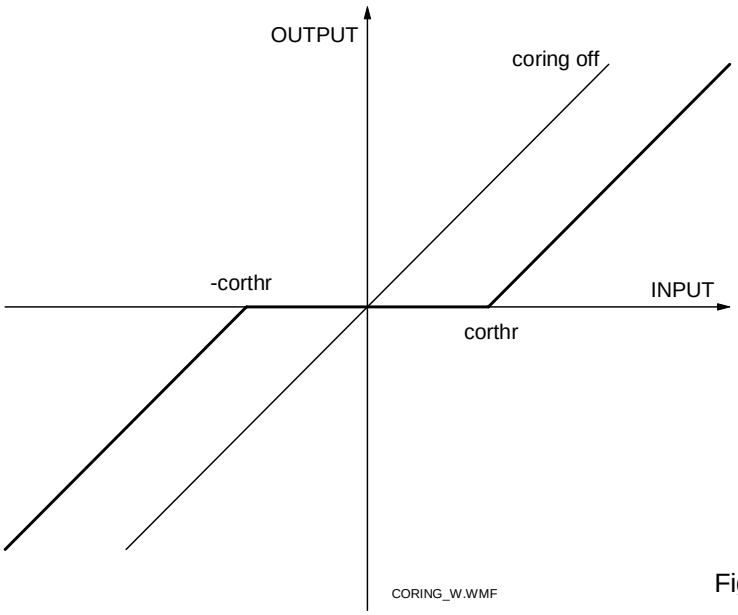


Fig. 56 Luminance coring

3/8 (parameter *NLP_L*) and μ = 0, 1/4 and 1/2 (parameter *NLP_U*). Preshoots are generated for μ = 0, symmetry is obtained for μ = 1/2.

In fig. 59 the transfer and group delay curves are given for the different combinations of λ and μ. In each plot the upper curves represent the group delay, the lower ones give the amplitude response. The left three plots show

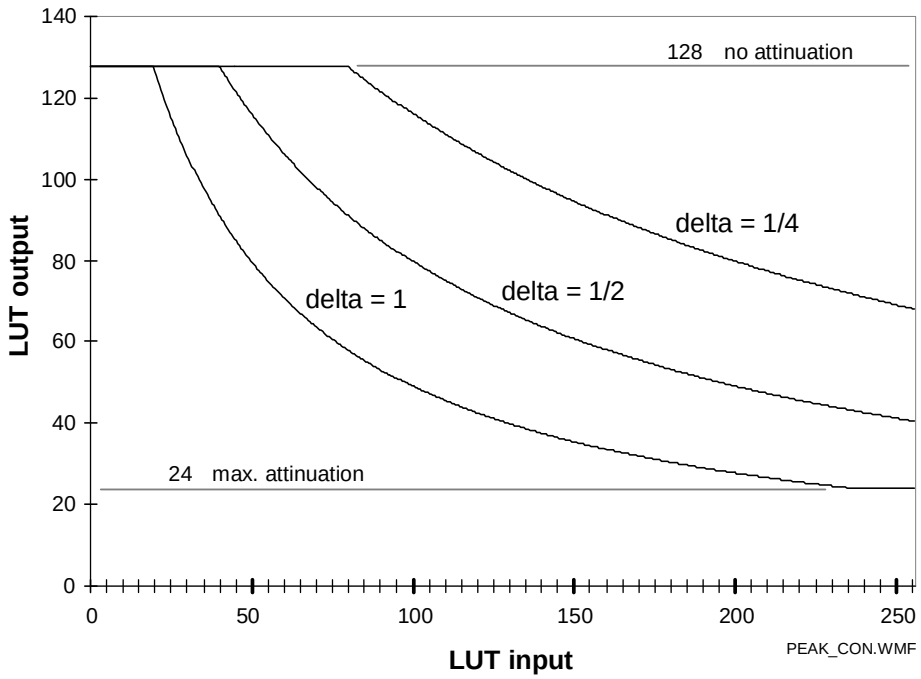


Fig. 57 Dynamic peaking control

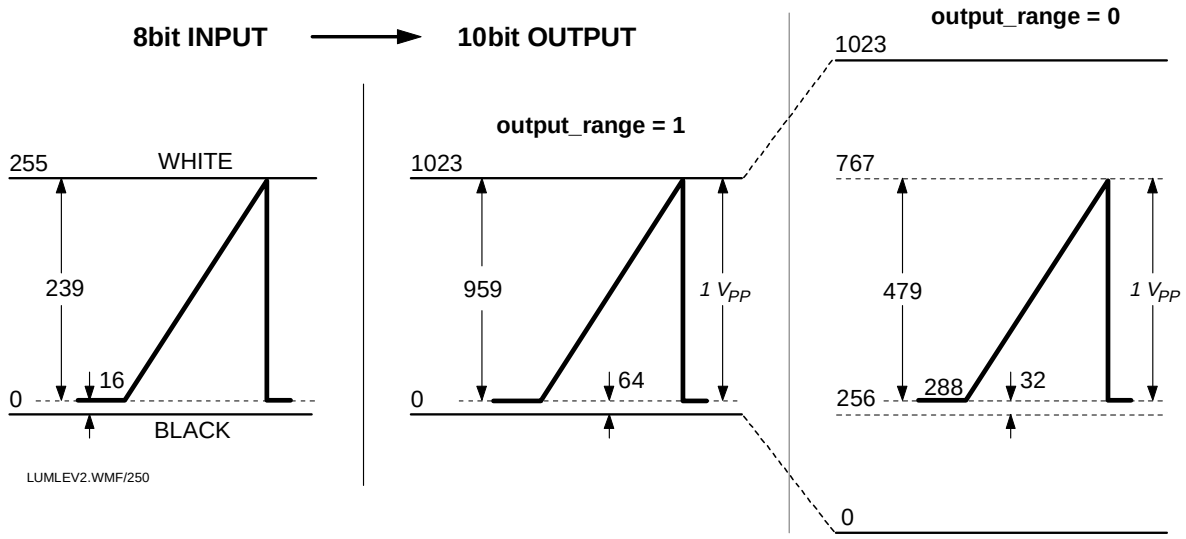


Fig. 58 Input / output signal levels of luminance signal

the behavior of the digital filter itself, the plots on the right side show the superposition of the digital filter and the analog postfilter.

The NLP filter has four settings of parameter *NLP_L* which give the following gain factors at 10 MHz:

Setting 3 almost completely compensates the sin x/x and postfilter loss at 10 MHz, if more gain is wanted then this should be done in the dynamic peaking block.

4.3.7 Post processing: borders, frames and blanking

In the post processing block borders and frames can be defined for various display options like

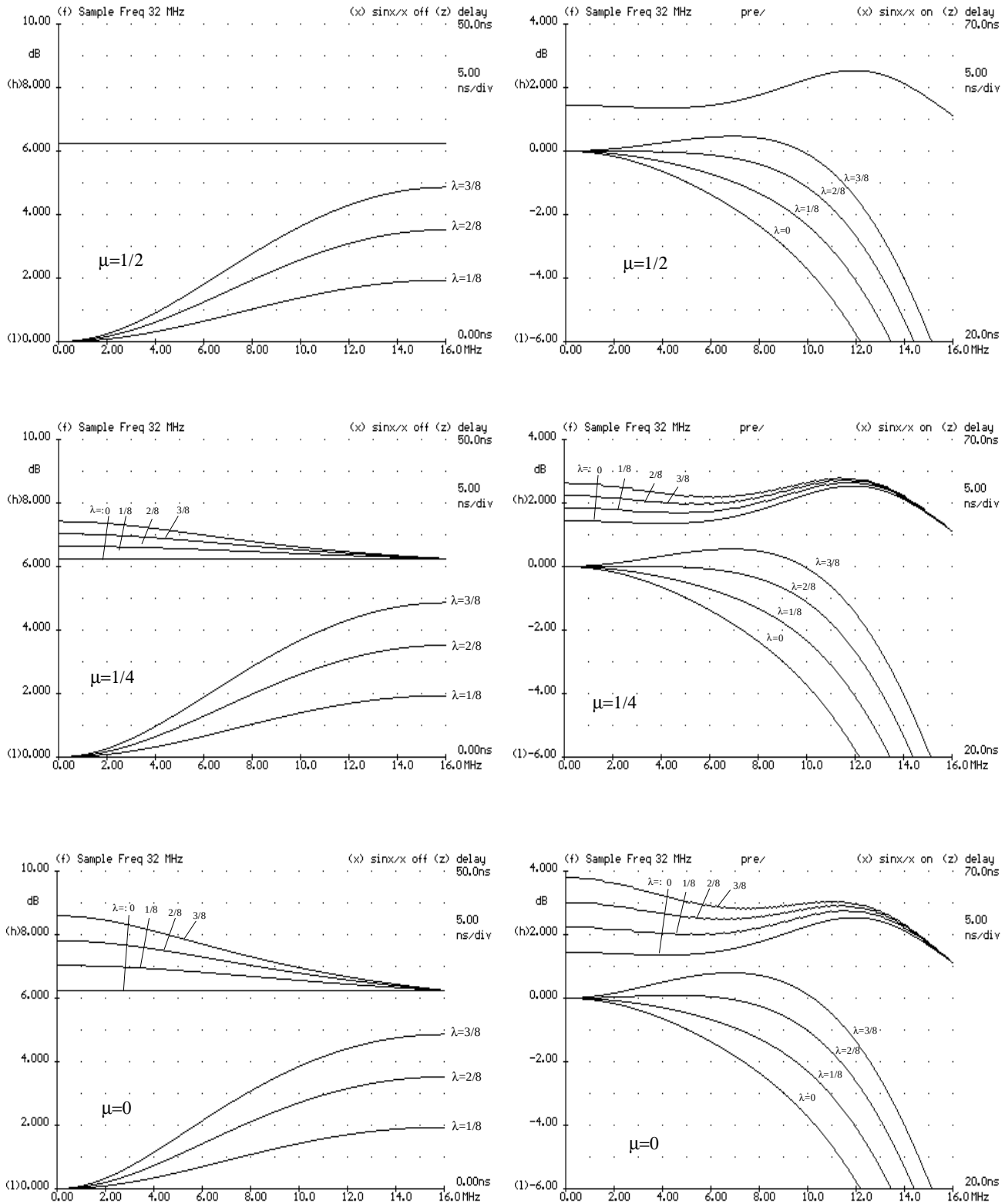


Fig. 59 Group delay and transfer curves of the NLP D/A filter

- side panels for 4:3 displays on a 16:9 screen

NLP_L	λ	gain [dB]
0	0	0
1	1/8	1.3
2	2/8	2.6
3	3/8	3.7

Fig. 60 NLP D/A gain settings

- window for POP (picture outside picture) on a side panel
- windows for PIP (picture in picture)
- frames for double window
- blanking to avoid border effects.

Fig. 61 shows the definition of borders. Borders start at position h_start and end at position h_stop, each value being defined in number of pixels. So if h_start > h_stop then two borders are generated at the left and right side of the screen. This case is used to define side panels, the register names are SIDEPANEL_HSTART and SIDEPANEL_HSTOP. If h_start < h_stop then a vertical bar on the screen is defined. This bar could be used as a separator in a double window display.

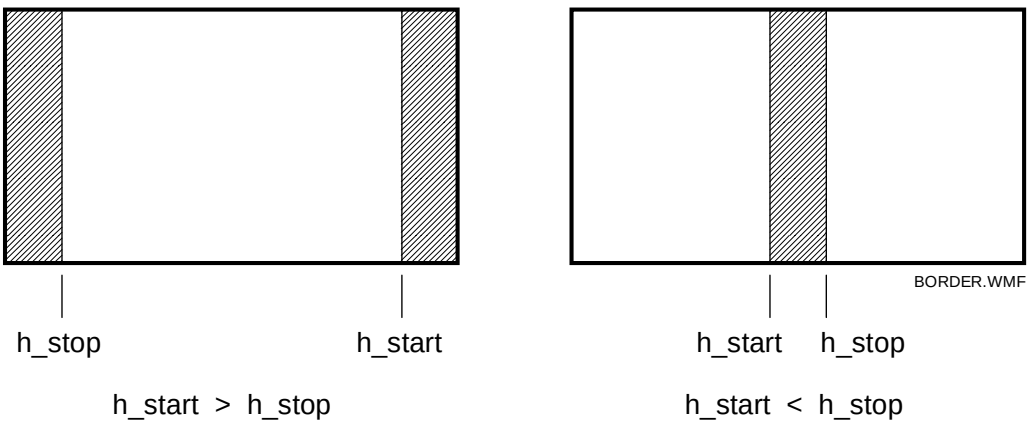


Fig. 61 Border definition

Fig. 62 shows the definition of frames. Like borders frames also start at position h_start and end at position h_stop, each value being defined in number of pixels. In vertical direction the beginning and end of a frame are defined by v_start and v_stop. With h_start > h_stop and v_start > v_stop and frame width and height set to zero this leaves a window on the screen like on the left side of fig. 62.

When only height is zero the vertical part of the frame is extended to the upper and lower edge. Shifted to the left or right side of the screen this sort of frame is suitable for POP (picture outside picture). Examples for various kind of frames are shown in fig. 63:

a) Side panels with surrounding blanking. The side panels can be applied at a 4:3 picture display on a 16:9 screen.
b) Large frame
c) Window, e. g. for PIP when no main picture is available.
d) PIP frame, e. g. display of a camera picture in the window within the main picture.
e) POP (picture outside picture), for example a 4:3 picture is displayed on the left side of a 16:9 screen, and the remaining gap is filled with a panel with a window for a second picture.
f) Frame with dividing bar in the middle, surrounded by a blanked frame, e.g. for double window display.

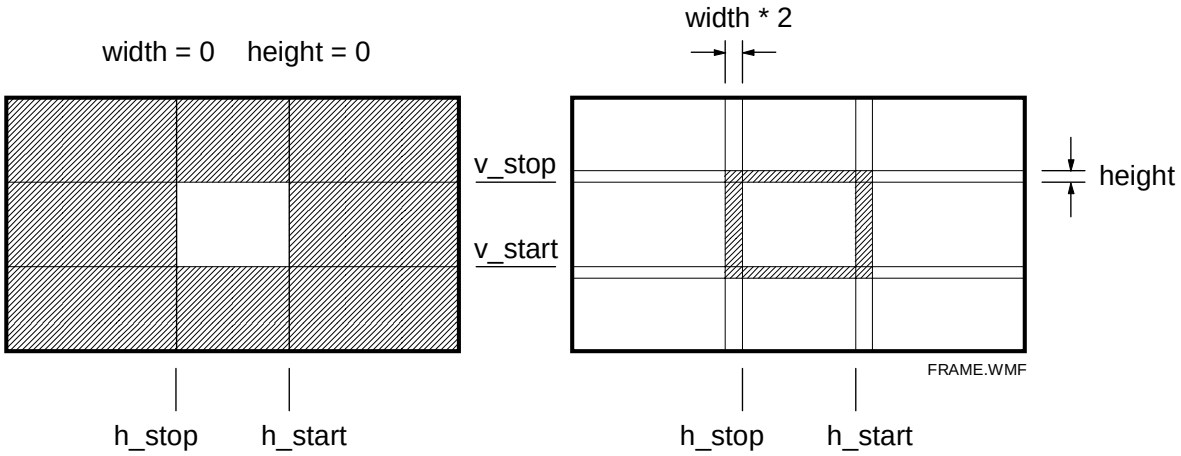


Fig. 62 Frame definition

During blanking intervals, the signals are set to nominal black values, i.e. 64 for the 10 bit luminance component and 0 for the signed 10 bit color difference signals U and V. For frames and borders a color can be defined. The unsigned 8 bit luminance value *SIDEP_Y* is internally multiplied by 4, and the signed 4 bit chrominance values *SIDEP_COLOR_U* and *SIDEP_COLOR_V* need a factor of 64 to become 10 bit signals.

4.4 Triple 10-bit digital-to-analog conversion

Three identical 10-bit digital to analog converters provide the analog luminance and color difference signals Y, R-Y and B-Y (YUV) signals. The output signals can be low pass filtered and eventually amplified outside the SAA4979. The nominal output levels for a 100/0/75/0 color bar signal are given in fig. 64.

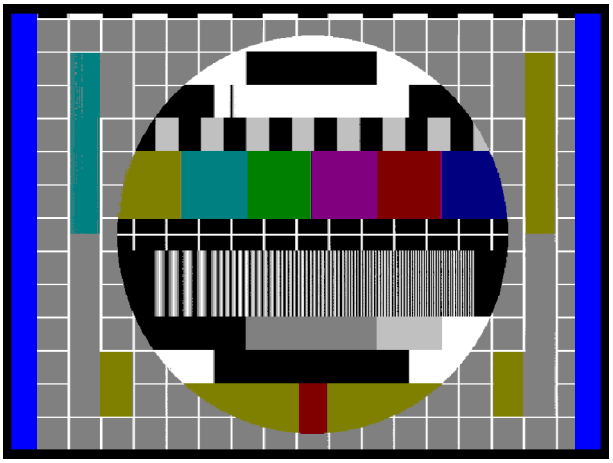
4.5 Microcontroller

The SAA4979 contains an embedded 8051 microcontroller core (µC) including 512 bytes of RAM and 32 kB ROM. It is placed on the display chip and runs on 16 MHz which is derived from the 32 MHz display clock. The microcontroller takes care of detailed processing of the various modes and presents as user interface a set of registers where modes can be set or data can be read. This register specification depends on the version of the on-chip firmware. In this application note the current version is described, which is version 4.4, see chpt. 10.

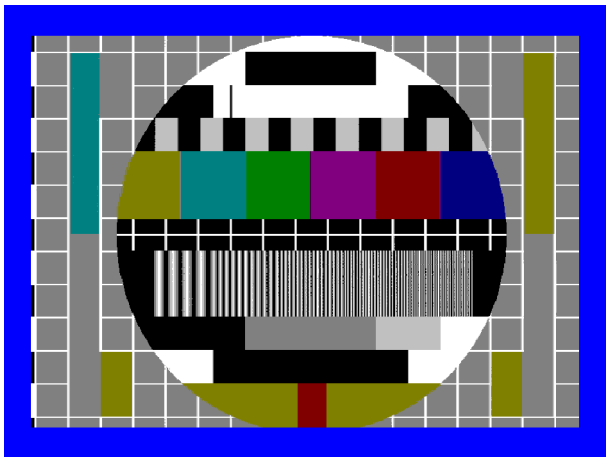
For communication with external ICs two serial busses can be used, the I^2C bus and the SNERT bus. The I^2C-bus interface is used in a slave receive and transmit mode for general communication with a central master microcontroller. Both standardized baud rates of 100 kBit/s and 400 kBit/s are supported. The I^2C bus address of the SAA4979 is 0110 100R/$\overline{W}$ (68_H/69_H). During slave transmit mode the SCL LOW period may be extended by pulling SCL to LOW (in accordance with the I^2C bus specification).

The SNERT bus is used for communication with slave ICs that also have this interface (like the SAA4992). It is a single master bus and uses the µC´s serial interface for transmitting and receiving data. Clock is supplied by pin SNCL while data is written or read through pin SNDA. These pins refer to the pins TxD and RxD of a standard 8051 µC, and the transfer mode is known as mode 0 of the serial interface. Address and data bytes are transmitted alternately. As reset signal the bus uses a third signal line (SNRST) to determine the correct address/data sequence as well as to update any readable registers in the devices. In a video environment however the vertical sync pulse is usually taken for this reset purpose, since SNERT transmissions are initiated by this pulse, too. The standard speed of the SNERT interface is 1 MBaud, but it can be set to 2 MBaud if the attached slave ICs support this data rate. [3]

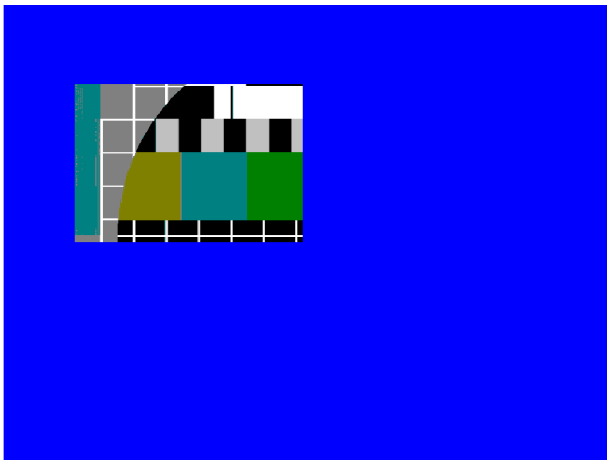
3. see also: Waterholter, Heinrich: The SNERT bus specification, Philips Semiconductors Application Note AN 95127



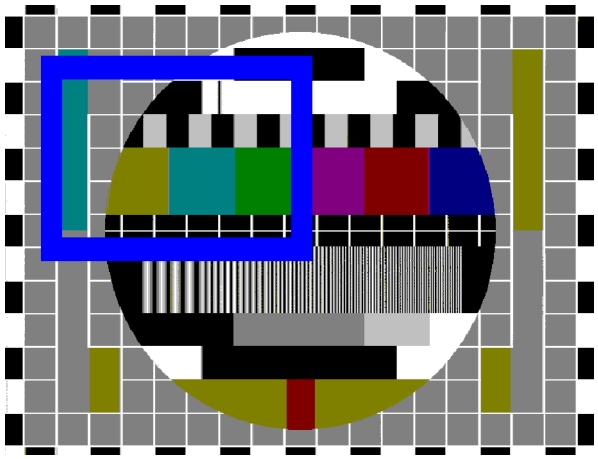
a) side panels and blanking



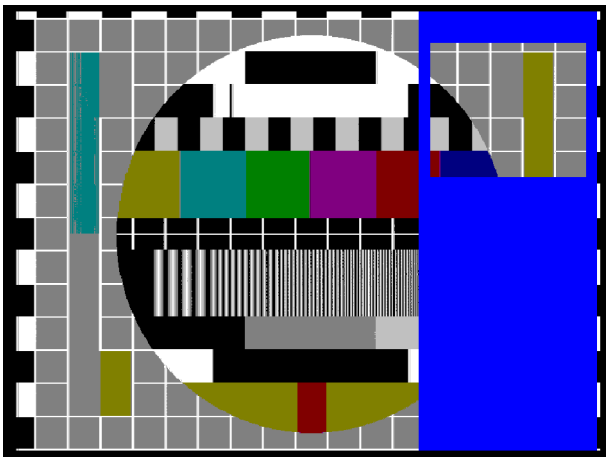
b) large frame



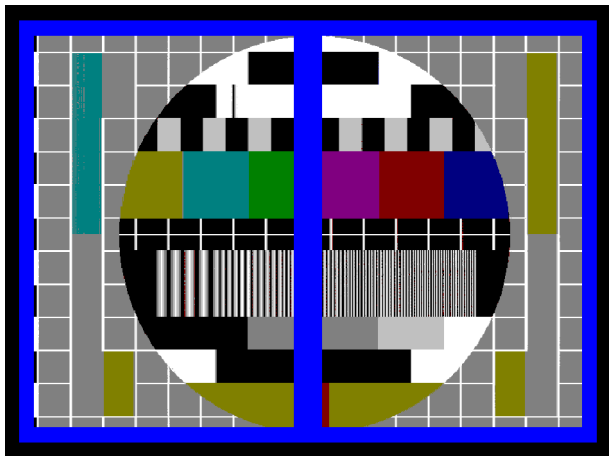
c) window



d) PIP window



e) POP and blanking



f) double window and blanking

Fig. 63 Examples for windows and frames

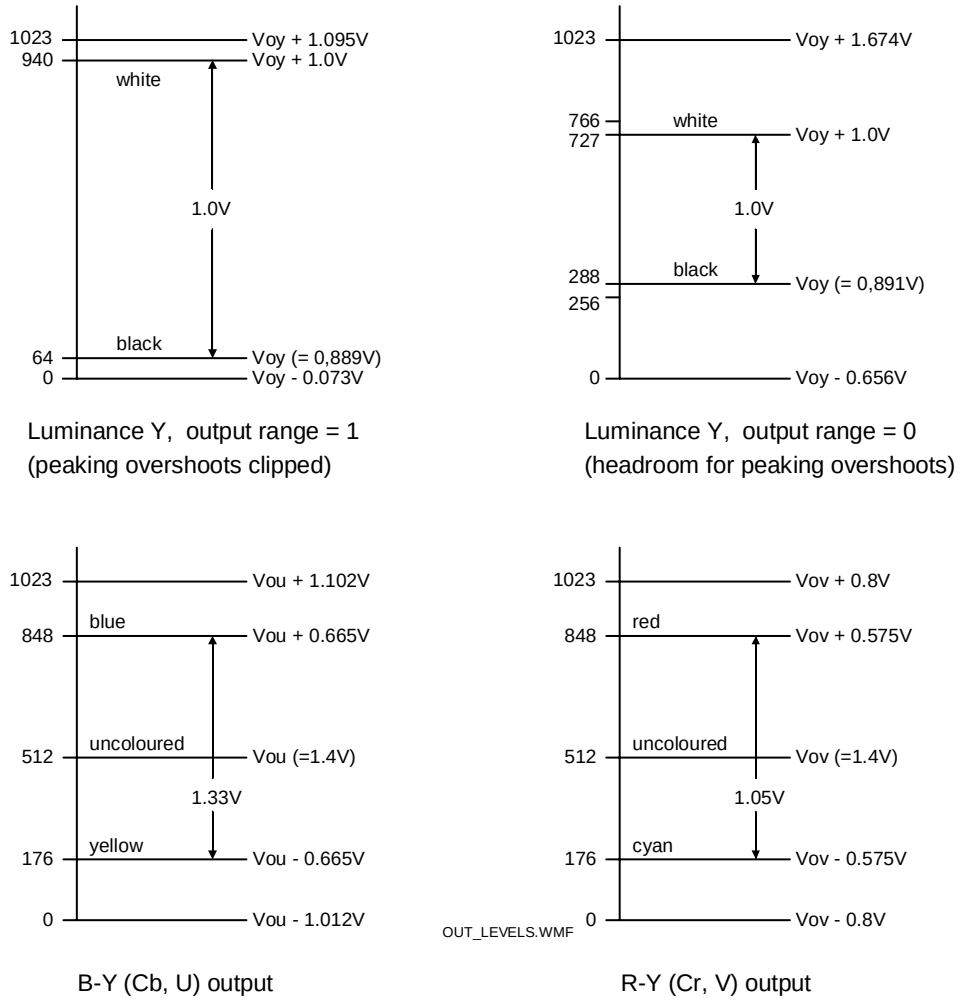


Fig. 64 Luminance and chrominance output levels

A parallel port (PORT 1) can be used for application specific signals. While pins P1.0, P1.6 and P1.7 are already used for SNRST and the I^2C bus signals SCL and SDA, the port pins P1.2 ... P1.5 are still available for specific purposes.

4.6 Memory controller

The internal memory controller generates the required control signals to operate the internal scan conversion memory. Its mode of operation is set by the microcontroller. Also the control signals (REO and IE) to run additional memories in applications with motion compensation ICs are generated. At pins HD and VD the horizontal display and vertical display pulses for the deflection power stages are generated.

The system controller also supports double window or picture-in-picture processing in combination with an external field memory by providing the required memory control signals (RE2, RSTW2 and OIE2).

4.7 Line locked clock generation

An internal PLL generates the 32 MHz line locked display clock CLK32. The PLL consists of a ring oscillator, DTO and digital control loop. The PLL characteristic is controlled by means of the microprocessor.

A 12 MHz crystal is used. Recommended values for crystal series resistance is <150 Ohm, parallel capacitance <7 pF and load capacitors are 12 pF and 18 pF, see fig. 65.

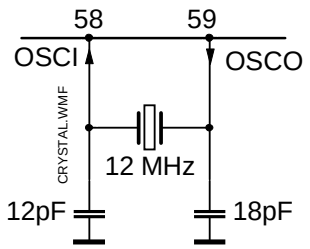


Fig. 65 Application diagram of Crystal for PLL

5. Functional description of the SAA4998

The key feature of the MK14EM module - motion compensated field and line rate conversion - is realized by the motion compensation IC SAA4998. It supports conversion to 100/120 Hz 2:1 (interlace) or 50/60 Hz 1:1 (progressive), an extensive list of features is given in table 1. Although not realized on the modules, the SAA4998 would also be able to generate 100/120 Hz 1:1 (progressive) by making use of the second output channel (Y_G and UV_G).

The SAA4998 or FALCONIC-EM (EM stands for *embedded memories*) is based on the SAA4993. The memories that the SAA4993 needs externally for field or frame storage are now integrated. Alternately they can also be used for PIP (picture-in-picture). Fig. 66 shows a block diagram of the SAA4998.

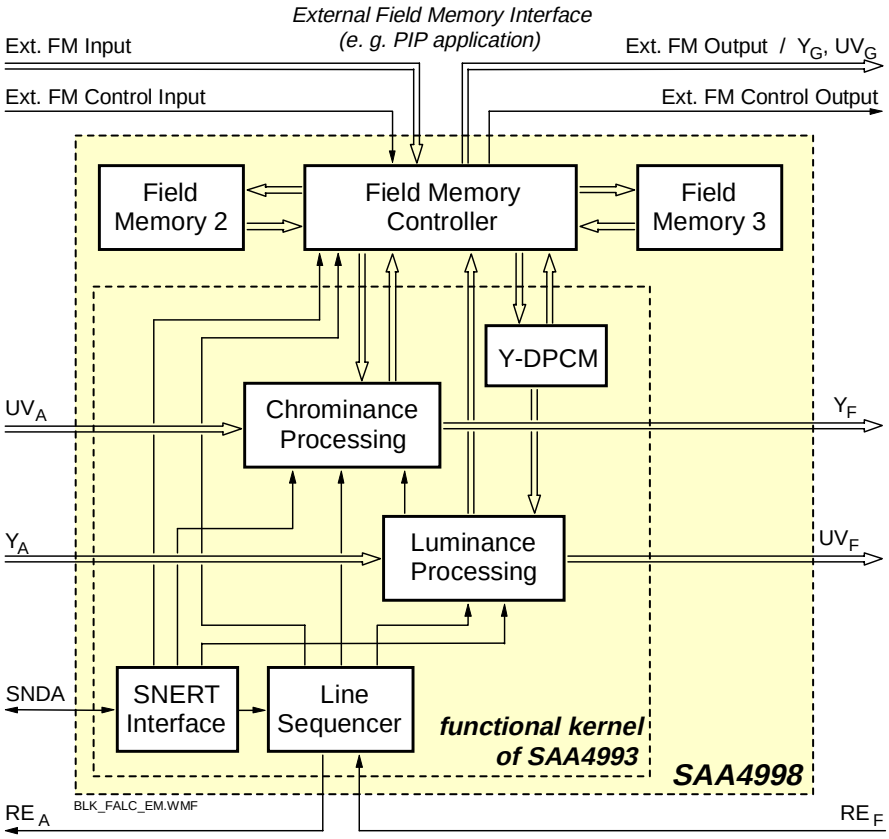


Fig. 66 Block diagram of the SAA4998

5.1 Problems in motion portrayal with picture rate conversion

The simplest approach to double the scan rate is to display each field twice. This eliminates large area flicker effectively but still has the problem of blurring or contouring of moving edges. This artifact is depicted in fig. 67. For a moving object it can be seen that its position is incorrectly represented in every second field. If the viewer tracks the object it is perceived double, as its location in every second field is not at the expected position.

Much worse is the display of movie material on a TV receiver or even in the cinema, because motion comes in a rate of only 25 pictures per second. On a 50 Hz TV each motion phase is displayed twice resulting in annoying jerky motion due to a lower picture update rate and therefore a larger position error between expected and displayed object position. In current 100 Hz TV each movie picture is repeated four times which still increases the jerkiness of the motion.

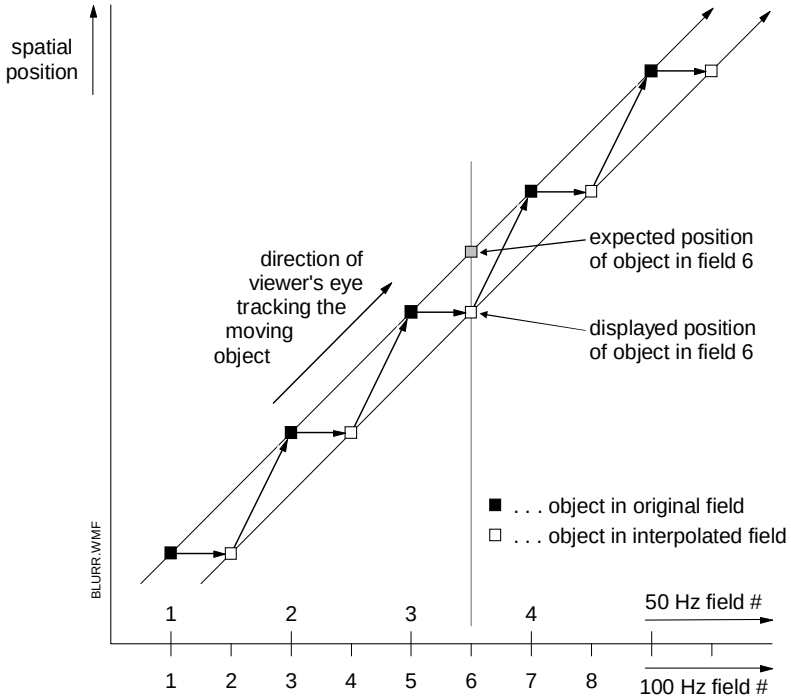


Fig. 67 100 Hz field repetition causes blurring at moving edges

5.2 Motion estimation and compensation for luminance

In order to overcome the above described problems a motion estimation technique is needed, so that objects in the interpolated image can be placed at the position expected by the viewer´s eye. The technique implemented in the SAA4998 is based on a 3-D recursive search block-matching algorithm. Fig. 68 demonstrates the block matching principle.

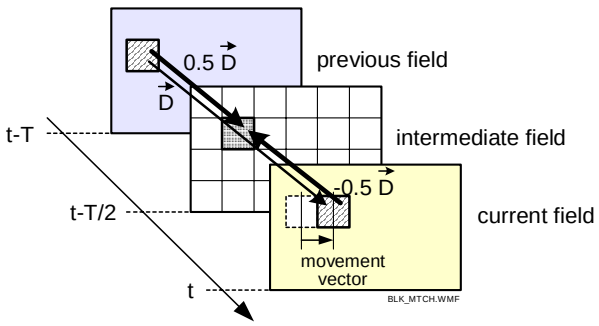


Fig. 68 Block matching principle

Motion estimation is performed in the luminance channel only. Motion compensated upconversion is done in the luminance channel while for the chrominance signal upconversion is done by a median filter. The vector range is ± 12 lines vertically, ± 31.75 pixels horizontally (sub-pixel accuracy). Motion estimation and compensation is done in the luminance channel only, a block diagram is show below (fig. 69).

5.2.1 Multi port RAM (MPR)

The multi port RAM (MPR) has the task to provide fast access to the contents of the current field and the stored frame. It consists of a number of line memories to hold the actual data, which are needed for processing.

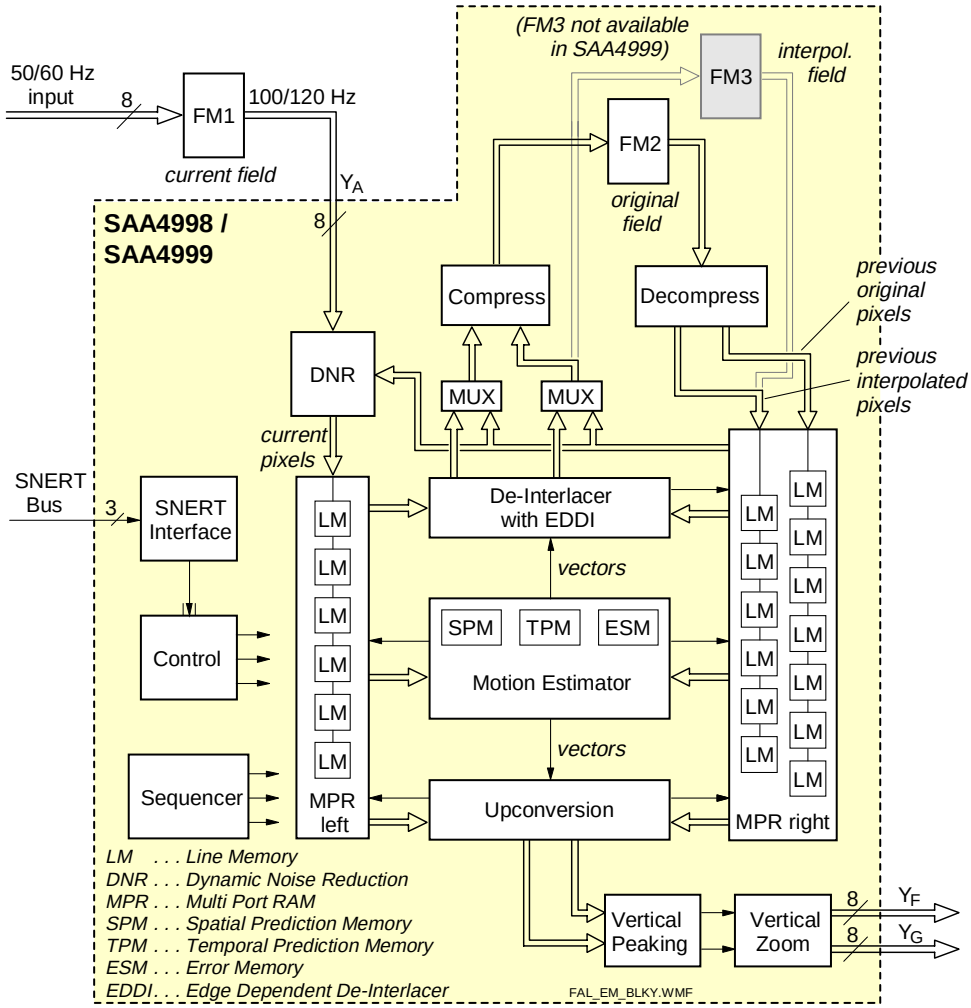


Fig. 69 Block diagram of the SAA4998/..99 luminance processing

In principle the MPR can be split into two main parts, the right memory tree and the left memory tree. The right memory tree consists of 13 line memories to hold 7 subsequent original lines from the previous field and 6 interpolated lines out of the previous field. This means, that 13 subsequent lines of a frame, generated from the previous field are available. The left memory tree contains 6 line memories to hold 6 subsequent lines of the current field, see fig. 69.

The blocks De-interlacer, Motion Estimator and Upconverter address the right and left part of the MPR with internally generated vectors. According to these vectors, the MPR provides the above mentioned blocks with their pixel data for the current field (left line memory tree) as well as for the previous (interpolated) frame (right memory tree).

The left memory tree is continuously filled with pixel data of the last 6 lines of the current field, which are first filtered in the DNR block. The right memory tree gets data of the interpolated previous field (field memory 3) and of the original previous field (field memory 2). Data of the current field are written back to the field memory 3 and field memory 2, respectively, after its processing (interpolation).

The SAA4998 can be run without FM3 (if this is used for other purposes like PIP). In this case the blocks Compress and Decompress are activated to reduce the amount of luminance data by a factor of 2 so it can be stored in one memory only. For compression and decompression DPCM (differential pulse code modulation) is used.

The SAA4999 does not have the FM3 available, so whenever storage of picture data in two field memories is required, compression and decompression is activated.

5.2.2 Motion estimator

The motion estimator calculates the motion vector of objects within an incoming video field by comparing the field itself with a previous frame. It reads pixel data from the current field and the previous frame via the local caches or multi port RAMs. Prediction vectors of the current and neighboring blocks which were estimated in the previous field period and stored in the Temporal Prediction Memory (TPM) are used as a basis for new estimations. From this information, it generates a new motion vector, which is again forwarded to the TPM, leading to a temporally recursive motion estimation.

The motion estimator works on a picture block size of 4 lines $\times$ 16 pixels, while a motion vector is assigned to a block size of 4 lines $\times$ 8 pixels in a checker board pattern (quincunx block subsampling), this means a subsampling of factor two, only every second block a motion vector is assigned to. For the other blocks the motion vector is interpolated. Fig. 70 show the principle.

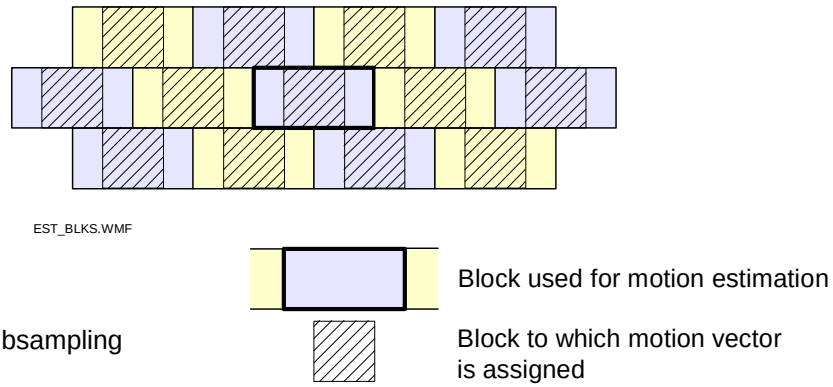


Fig. 70 Motion estimator block subsampling

The main basis for finding the movement vector of a block are the vectors of the neighboring blocks (spatial prediction vectors) and the vectors of the current and neighboring blocks of the previous field (temporal prediction vectors). This situation is depicted in fig. 71.

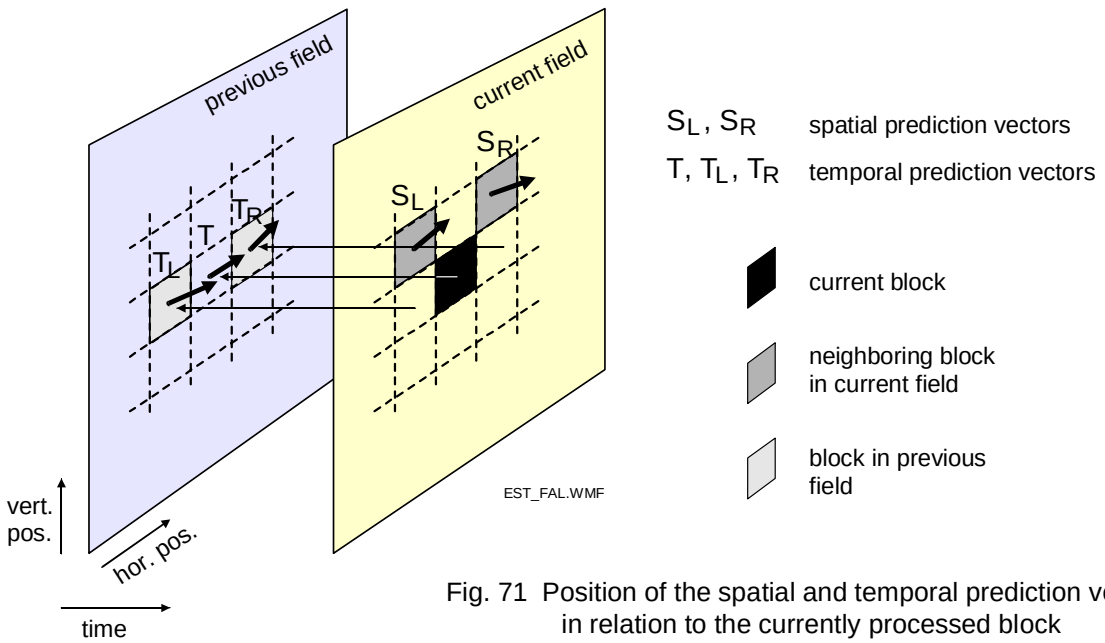


Fig. 71 Position of the spatial and temporal prediction vectors in relation to the currently processed block

Besides the spatial and temporal prediction vectors other possible vectors are also taken into account. The complete set of candidate vectors consists of the following:

- T (current temporal vector): current vector interpolated in the previous field for the actual block
- T_R (right temporal vector): vector right of the current vector in the previous field
- T_L (left temporal vector): vector left of the current vector in the previous field
- C_{max} (max vector): maximum vector of a certain area calculated in the previous field
- 0 (zero vector): vector 0 $\rightarrow$ no motion
- S_L (left spatial vector): left side neighboring vector in the current field
- S_R (right spatial vector): right side neighboring vector in the current field
- C_P (programmable vector candidate): selected by block candidate selection.

For each picture block four candidates will be selected as candidate vectors. The selection of the vectors is programmable, an example is depicted in table 2. U represents a random update vector which can be applied to any prediction vector, see fig. 72.

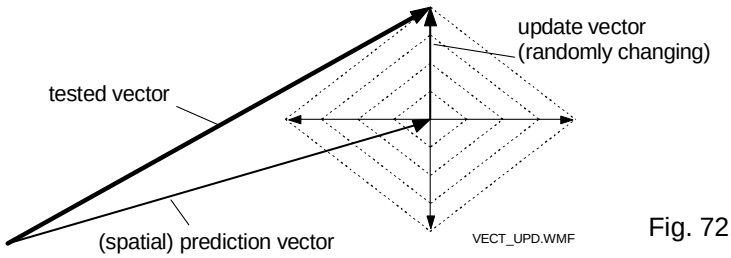


Fig. 72 Recursive search trying to find a better vector

Candidate Number	Block Number	Estimator	PAN_ZOOM reliable?	Selected Candidate
1	-	L	-	S_L
	-	R	-	S_R
2	-	L	-	T_R
	-	R	-	T_L
3	odd	L	-	S_L + U
		R	-	T + U
	even	L	-	T + U
		R	-	S_R + U
4	odd	L	yes	C_P
			no	C_{max}
	even	L	-	0
	odd	R	-	0
	even	R	yes	C_P
		R	no	C_{max}

Table 2 Selection of vector candidates

C_{max} defines the maximum candidate within a certain area around the current block B. It is found by scanning 5 vectors around B as shown in fig. 73. This maximum vector changes from block to block. So every time C_{max} is a candidate the 5 motion vectors are evaluated. This maximum candidate enables fast convergence of motion vectors.

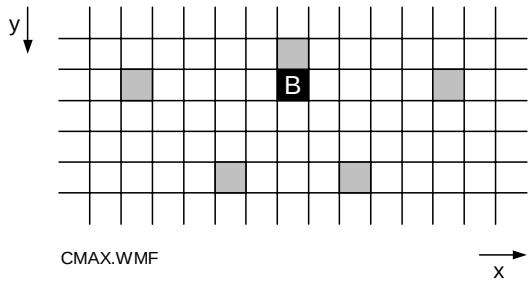


Fig. 73 Selection of C_{max}

C_P is a programmable vector. This candidate is possibly selected when camera panning or zooming is detected, see fig. 74. For this the vectors (V_0 .. V_8) of nine blocks are constantly read by the external microprocessor, and if size and direction indicate a panning situation this vector is written back to the SAA4998. If zooming is also detected the delta-value in x- and y-direction is also written back. From these four basic values the SAA4998 will construct a vector field for all blocks in the picture by linear interpolation.

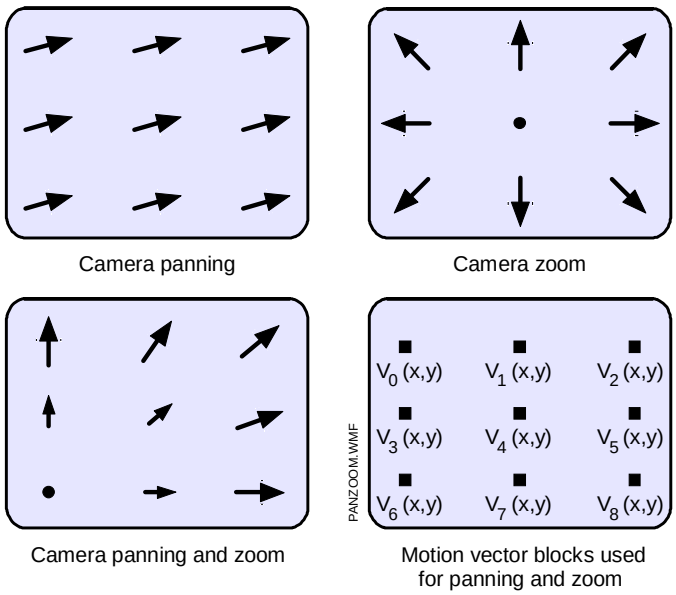
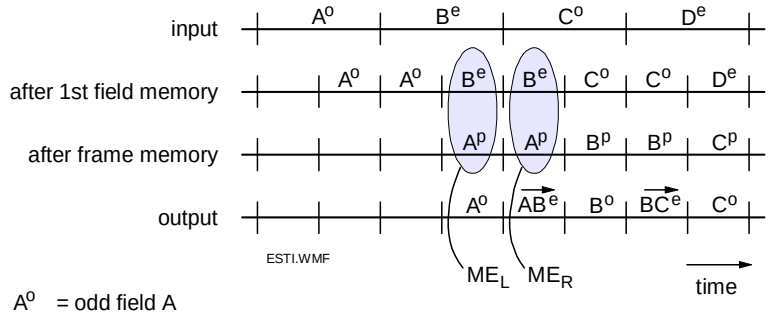


Fig. 74 Motion vectors for panning and zooming

For every input field motion estimation is done twice, but for different candidate sets. The first motion estimation is called left (L), the second is called right (R), which is marked in the column 'estimator' in table 2. In hardware only one motion estimator is used, which is multiplexed in time. In field rate doubling ($2f_V$ output) two estimations can be done for every input field, see fig. 75. The left estimator is used in the odd output fields and the right estimator in taken for the even output fields. In $1f_V$ output mode (progressive scan) the use of the estimators depends on the zoom factor: with a vertical zoom factor >= 1, the use of the left and right estimator toggles per output line, with a vertical zoom factor below 1 (compression up to a factor of 2), it is only possible to use either the left or the right estimator per field (this will reduce the estimator performance, as only 50 % of the estimations can be done).



A^o = odd field A
A^p = progressive scan frame A
$\overrightarrow{AB}^e$ = motion compensated interpolated even field AB
$ME_{L, R}$ = motion estimator left, right

Fig. 75 Two estimations per input field

The vector candidates define a translation from field t to t–T. If the intermediate field is the point of reference, the displacement is equivalent to half the motion vector in both directions. Therefore the candidates are split in two parts, see fig. 76. In order to prevent that the vectors point to information that is not available (due to interlace or subpixel accuracy), they are ‘rounded’ to the nearest original data. These split vectors are used to address the pixel data in the current field and in the previous field. The relevant lines of these fields are located in the multi port RAMs (MPR).

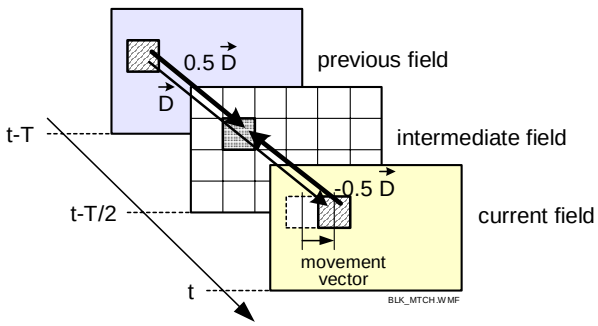


Fig. 76 Split vectors

For every candidate the Sum of Absolute Differences (SAD) is stored in an error memory (ESM). The candidate that delivers the smallest SAD could be considered as the best fitting candidate, and therefore, the best motion vector. But however, in some cases some candidates might be preferred above others. Therefore a (programmable) penalty can be added to each vector increasing the SAD. Finally the least error is calculated and the associating vector index (least error index) is determined. The least error index controls which vector will be put forward, via an additional temporal filter, to the temporal prediction memory (TPM).

5.2.3 Temporal prediction memory (TPM)

The temporal prediction memory (TPM) stores the vectors which are calculated by the motion estimator, one for each block (4 lines, 8 pixels). Due to block subsampling (see fig. 70) only for every second block a motion vector is stored. A 4k x 16 bit SRAM, inside of the IC, is capable to store the vectors of a complete field (848/16 vectors/line, 292/4 + 3 vertical blocks, results in 4028 words). When reading from the memory the TMP generates

interpolated vectors for the blocks for which a vector was not stored. The median filtering is done according to the following rule:

$$i = \text{median}\left(a, k, \frac{b+h}{2}\right)$$

i: interpolated vector
a: vector left of the current block
k: vector right of the current block
b: vector below the current block
h: vector above the current block

The vectors stored in the memory and the interpolated ones are not used directly in the upconverter and progressive scan converter since this might result in some noticeable blocking artefacts. Therefore the vectors are further interpolated down to a block size of two (horizontal) pixels. This algorithm is called block erosion and is done in two steps:

In a first step the motion vector block is split into 4 quadrants. The four corresponding motion vectors D_{00}, D_{01}, D_{10} and D_{11} are found by median filtering of the block itself and the horizontally and vertically adjacent blocks, this yields a vector for each block of 2 lines by 4 pixels. In a second step the process is repeated further resulting in a block size of 2 (horizontal) pixels, see fig. 77.

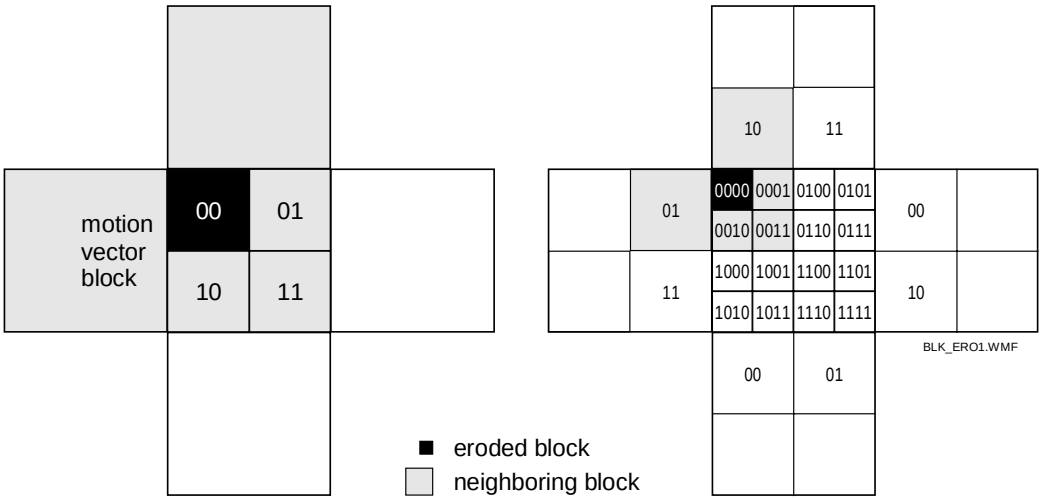


Fig. 77 Block erosion

5.2.4 Deinterlacer

The task of the deinterlacer [also called PROgressive scan conversion (PRO)] is to convert the interlaced input signal to a progressive one. The deinterlaced picture data is stored in a frame memory (FM2 / FM3). In the deinterlacer the missing lines of the incoming field are interpolated and the previous field is motion compensated. Data in the frame memory is shifted by the motion vector towards the data in the input field.

In comparison to the SAA4993 the deinterlacer block has been expanded and improved by the block EDDI which stands for 'edge dependent deinterlacing'. This block analyzes the progressive output signal of the standard deinterlacer and in case of staircases along an edge replaces pixels to generate a smooth edge, see fig. 79.

The inputs for the deinterlacer are the incoming field from the chip internal left memory tree of the multi port RAM (MPR) and the PRO output data, that was already stored in the frame memory and read into the right line memory tree of MPR (from PRO previously converted field). The motion vectors from the temporal prediction memory (TPM) are used to do a motion compensation for the image data. The output is again stored in the (external) frame memory and thus used by the upconverters. So the general functionality of the deinterlacer is to add the interpolated lines to the original incoming lines (field in) and to mix them with motion compensated frame data.

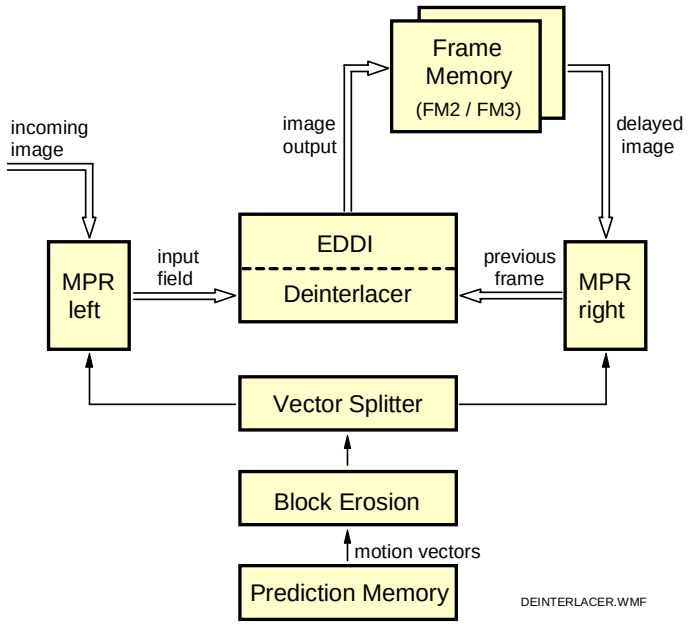
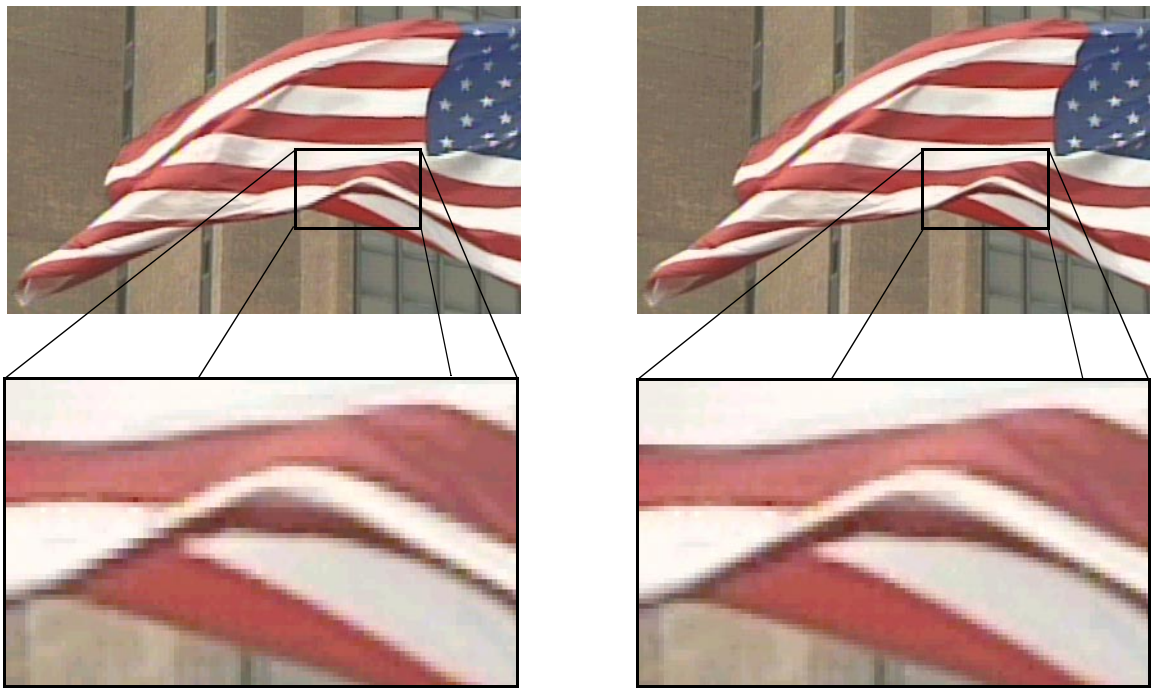


Fig. 78 Deinterlacing with EDDI



EDDI off

EDDI active

Fig. 79 Removing deinterlacing artefacts with EDDI

The deinterlacer works in two modes, the normal (video) mode (pro_mode = 1) or the embrace (movie) mode (pro_mode = 0).

Movie mode is the simpler one of the two. Because both fields stem from the same piece of film there is no motion between them and the data is only passed through.

In video mode the functionality is more complex. Data from the two original lines are taken to interpolate (median filtering or averaging) an intermediate value for the missing pixels in the incoming field. Together with pixel data from the previously interpolated frame and the motion vector three motion compensated intermediate pixel values are calculated. The differences between these intermediate 'right' (previously interpolated frame) pixel and the intermediate 'left' (original field) pixels are calculated, postprocessed by different factors and then output as original and interpolated line data to the field memories FM2 / FM3.

The progressive lines from the standard deinterlacer are evaluated by the block EDDI. This block is based on an algorithm which searches for edges in the picture and determines angle and length of the transient. In cases where the standard deinterlacer has generated staircases this block will replace the badly interpolated pixels by the ones from its own calculation. The improvements are visible in fig. 79.

5.2.5 Upconverter

The upconverter converts the incoming field frequency to the selected output field frequency, it acts as field and frame rate converter.

The quality of field rate conversion improves significantly with motion-compensation techniques. It becomes possible to interpolate new fields at their correct temporal and spatial position. This results in smooth motion portrayal without loss of temporal resolution.

However, as motion vectors are not always valid for every pixel or object non-linear filtering is used in order to minimize visible artefacts. The algorithm generally consists of two steps. First the displacement of objects within successive fields of an image sequence must be determined; a motion estimator is needed for this. Secondly, the resulting motion vectors of the first step are used to interpolate new image fields in between existing ones, this is done in the upconverter.

Fig. 80 shows the block diagram of the upconverter. The vector splitter gets a motion vector from the temporal prediction memory (TPM), processes it and applies it to the left and right cache (multi port RAM, MPR). The MPR returns the actual pixels as well as the delayed pixels of the requested positions as an array of pixels. These pixels are interpolated with a non-linear filter (cascaded median filter) to form the pixels of a lower and an upper (de-interlaced) line which are output to the zoom circuit. A circuit called "egg-slicer" is used to verify motion detection.

Vector splitter

The motion compensation range is six lines up and six lines down as well as 16 pixels to the left and 16 pixels to the right. The vectors generated by the motion estimation and stored in the TPM are used by the vector splitter to calculate the address of a set of pixels out of the left and the right multi port RAM (MPR) needed by the upconversion circuit. The lower bits of these vectors are used by the sub-pixel interpolation circuit. The final accuracy is 0.25 pixel.

E. g. in the standard application of field rate doubling in video mode a new field has to be generated half-way between the current and the delayed image. In this case the vector splitter is set to 0.5, see also fig. 76 on page 64. For different field rate conversion factors the vector splitter would need changing interpolation factors from field to field ranging from 0 = current field to 1 = previous field.

Upconversion circuit

In the upconverter the vector shifted pixels from the left and right cache are taken to generate the pixels for the intermediate field. Two of these circuits are available to generate a lower and an upper video line for the zoom circuit.

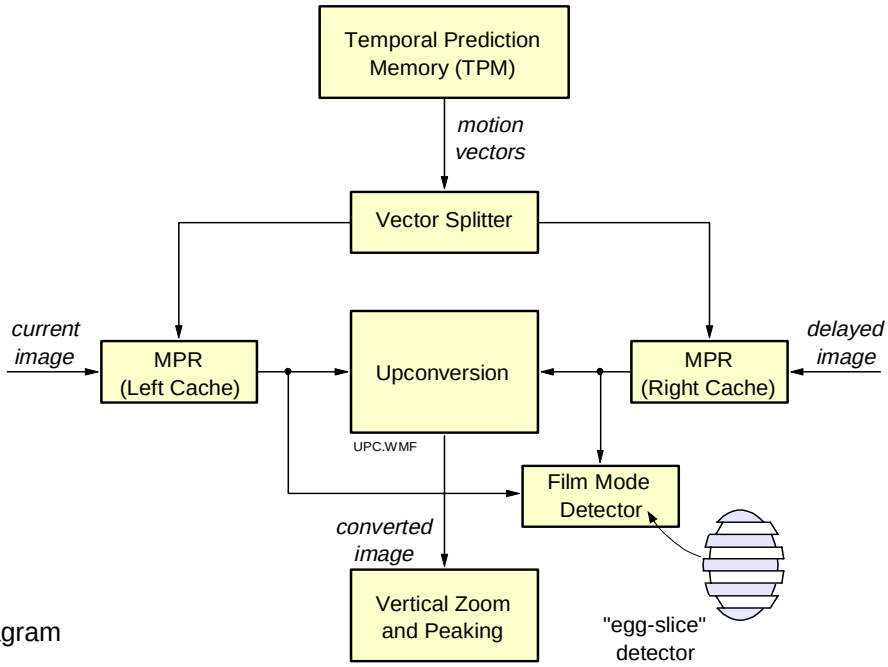


Fig. 80 Upconverter block diagram

Film mode detector

The film mode detector (also called “egg-slice” detector) calculates the difference between the unfiltered inserted field and the result that a motion adaptive field would give by calculating a median and sum of absolute difference over both fields. The results can be read by the microprocessor and be used to detect or verify the correct processing mode.

5.3 Vertical Peaking and Zoom

The block vertical peaking and zoom provides a frame based vertical (programmable) peaking and a vertical zooming algorithm for the luminance signal.

From the upconverter two pixels are delivered to the block synchronously: one of the lower and one of the upper video line. Both pixels are processed in the peaking circuit, which has programmable coefficients to select anything from a strong peaking to a strong smoothing. The diagram in fig. 81 gives the transfer function of the peaking filter. Some of the 16 possible peaking coefficient steps are outside a usable range and not specified; the diagram gives the ones that can be chosen.

From the peaking circuit both data streams are fed to the vertical zoom circuit. Here the picture is resized in vertical direction by linear interpolation between the two data streams. Any parameter from a compression factor of 2 to an expansion factor of 256 can be set.

There are two output busses and two output modes available: in standard mode the signal is output on bus F only and bus G is switched to high impedance. In matrix mode both busses F and G are used. In this mode a $4 f_H$ video output can be generated. Bus G displays a higher position of the video compared to bus F, bus G has a vertical offset of 1/2 a line step with respect to bus F.

In the SAA4998 the bus G output is combined with the external memory output. So if this bus is used in PIP or double window mode, matrix mode is not available. In this case two SAA4998s would have to be used, one for PIP and one for motion estimation.

In matrix mode the peaking function should not be used as it will be undefined.

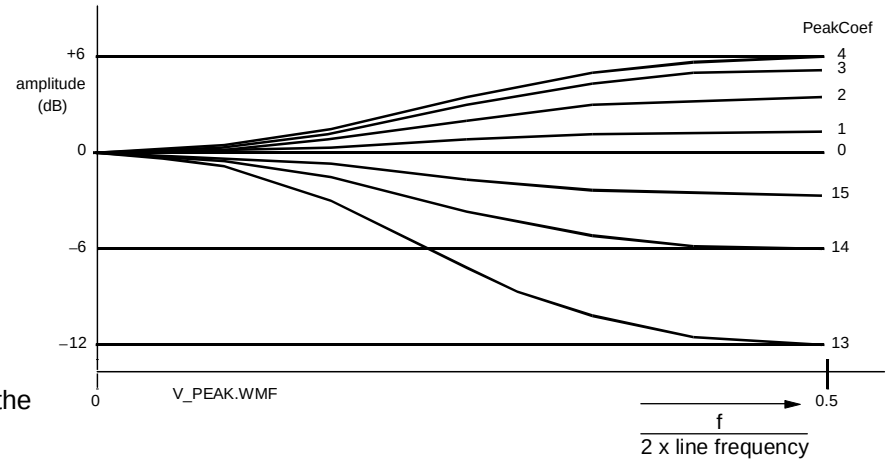


Fig. 81 Frequency response of the vertical peaking function

5.4 Chrominance processing

Basically the chrominance data path is similar to the luminance data path. However no motion estimation is done here. When motion vectors are needed they are taken from the luminance motion estimator. A block diagram of the chrominance processing is shown in fig. 82.

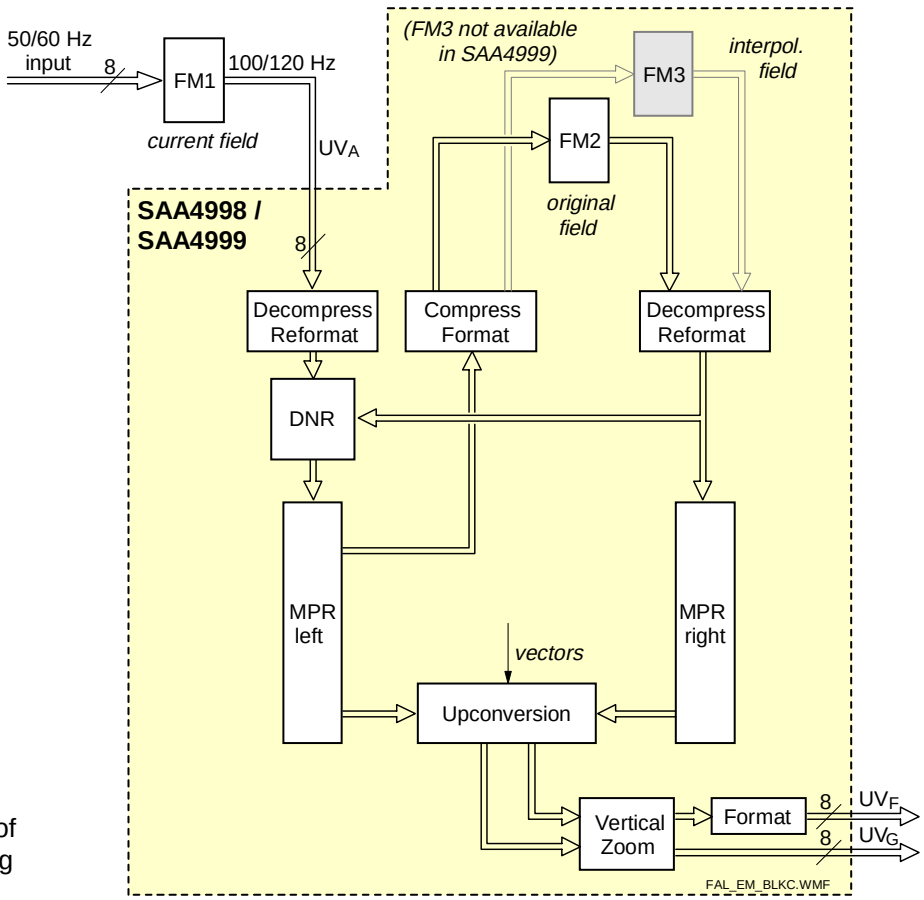


Fig. 82 Block diagram of chrominance processing

A special feature of the chrominance part is the color vector overlay mode. Instead of displaying the upconverted colors, the local motion vectors are overlaid as colors on the upconverted luminance. The horizontal components of the vectors are displayed as U, the vertical components as V. This overlay mode can be used for evaluation of the motion estimation process, but it is also well suited for demonstration purposes.

5.5 Memory configuration

The SAA4998 can operate with two or three field memories. With three field memories the maximum performance is achievable. The first one (FM1) is the scan conversion memory which is located in the SAA4979. Field memories FM2 and FM3 make up the background frame memory for motion compensation, they are inside the SAA4998 and contain frame data for luminance, but only field data in 4:2:2 format for color. FM2 holds the luminance of the previous field while FM3 holds the interpolated lines, together this gives a progressive image in the memories. In chrominance the 8 bit data in the 4:2:2 format are split up in 4 + 4 bits for each of the memories, so only the data for one field can be stored.

The memories in the SAA4998 have a separate input bus and therefore can be used for purposes other than motion compensation, like PIP or Double Window. In these applications a buffer memory is needed to compress the picture and synchronize the data stream of the subchannel to that of the main channel.

The SAA4998's embedded memories can be configured to work in three different modes which are selectable by software via SNERT interface (see table 3). The state of the input pins PIPON and TWOFMON will define the default functional mode after power-up or reset. These setup values are only presets and will be held in the register until the SNERT host (e.g. SAA4979) has altered these register settings.

PIPon	TwoFMon	Function
0	0	Both internal field memories are in use by the motion compensation function
0	1	not allowed
1	0	Motion compensation works with reduced data rates (YUV 4:1:1 or YUV 4:2:2 DPCM) and uses only field memory 2; field memory 3 is switched to the external PIP port. The field memory 3 will support low resolution PIP mode in this case
1	1	Motion compensation is disabled and both field memories are switched to the external PIP port.

Table 3 Application Modes

- **Full motion compensation** (PIPon = TwoFMon = 0)
  If both memories are used for motion estimation and compensation, these functions work with the highest performance. In this case no second channel can be displayed because no PIP buffer memory is available.
  The external ports for the PIP field memory access will have no function in this mode. Instead the output bus G is available as second output for 100 Hz data.
- **Low resolution PIP** (PIPon = 1, TwoFMon = 0)
  If the second input channel is activated to display picture-in-picture in low resolution mode, FM3 is used as buffer memory (max. 190 000 pixels per frame in YUV 4:2:2 mode including ITU control signals) and is therefore unavailable for motion estimation and compensation. This configuration also uses FM1 as first field memory but takes FM2 either for storage of only one field or for storage of a compressed frame. Compression is done by a factor of 2 both in luminance and chrominance, whereby a 2-dimensional DPCM[1] is used in luminance and a 1-dimensional DPCM in chrominance. So in this 4:2:2 DPCM mode

1. DPCM: Differential Pulse Code Modulation

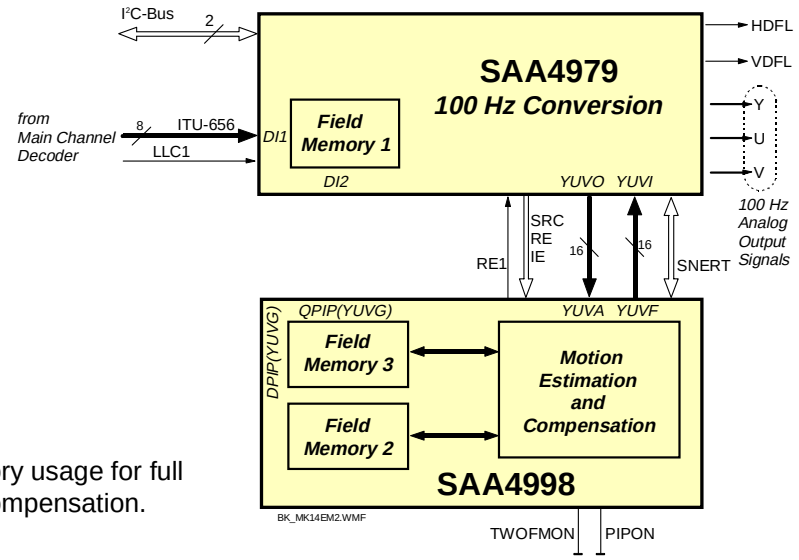


Fig. 83 Data flow and memory usage for full motion estimation/compensation.

again the luminance of one frame and the chrominance of one field can be stored.
In the SAA4998 FM3 is unavailable whenever one memory is needed for PIP/Double Window purposes.
In the SAA4999 the third field memory is generally unavailable.

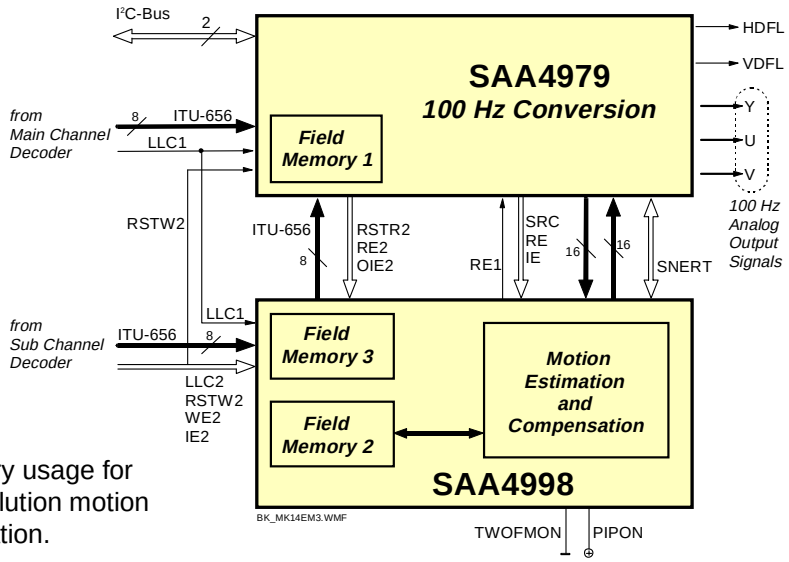


Fig. 84 Data flow and memory usage for half PIP and low resolution motion estimation/compensation.

- **Full resolution PIP** (PIPon = 1, TwoFMon = 1)
  Both memories are used as PIP buffer with a resolution of about 380 000 pixels per frame in YUV 4:2:2 format. In this case no motion estimation and compensation is available.
- **Full resolution PIP and full motion compensation**
  If full resolution PIP as well as full performance motion estimation and compensation is required, then two SAA4998 have to be used as shown in fig. 86. The one used for PIP is not connected to the SNERT bus and needs the pin setting PIPON = 1 and TWOFMON = 1. The chip used for motion compensation should be preset to PIPON = 0 and TWOFMON = 0.

Bus F, the main output, is 16 bits wide and can output data in 4:2:2 format. Alternately, also the 4:1:1 format can be selected.

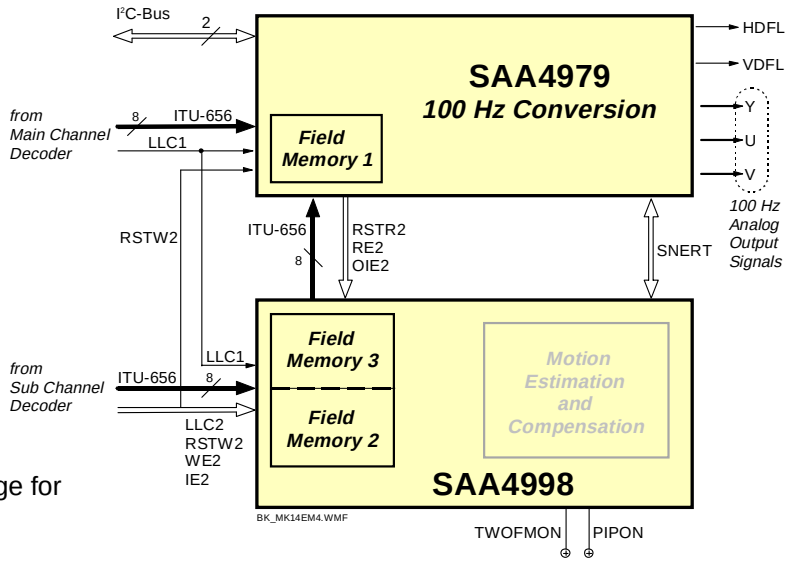


Fig. 85 Data flow and memory usage for highest resolution PIP.

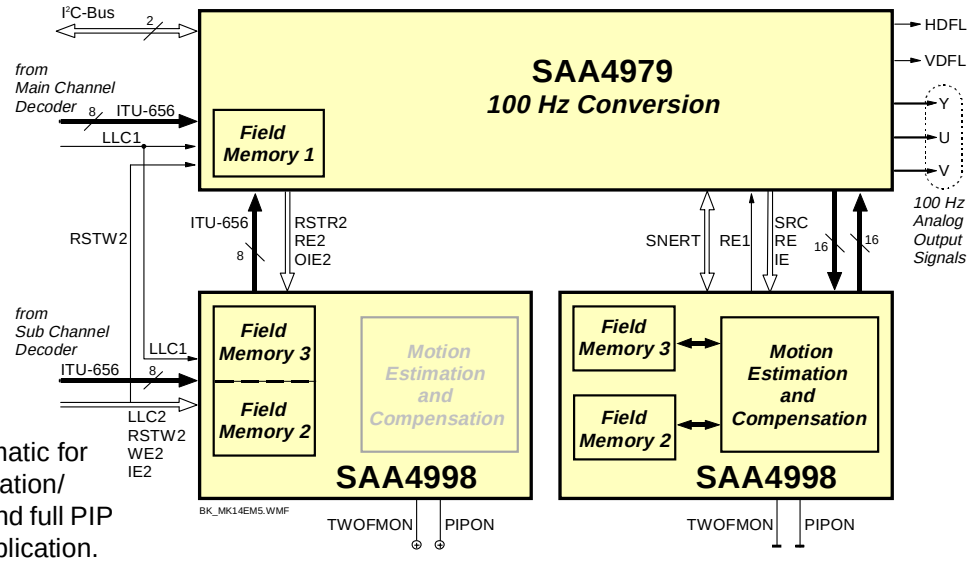


Fig. 86 Simplified schematic for full motion estimation/ compensation and full PIP performance application.

5.6 100 Hz progressive display

In applications where a 100 Hz progressive display (64 kHz line frequency, $4\,f_H$) is wanted, bus G can be activated. It is also 16 bits wide and outputs data in the 4:2:2 format. Both outputs each generate picture data at a rate of $2\,f_H$, so two external line memories are needed for doubling the data rate and multiplexing to a single $4\,f_H$ output. This application is not supported on the module MK14-EM.

5.7 Dynamic Noise Reduction (DNR)

As the block diagrams in fig. 69 and fig. 82 show, dynamic noise reduction is performed right after the signal is input from bus A. The current pixels and the ones from the previous field at the same location are compared and mixed.

There are two operating modes available: the user controlled mode and the signal adaptive mode. In the user controlled mode a fixed (user defined) ratio is set for averaging the new and old pixel data. This mode can easily lead to smearing effects in moving pictures and scene changes and therefore should not be used for normal operation. The ratio (also called the *k-factor*) is defined by means of a control register. A k-factor of 1 means total recursion and results in a still (frozen) picture.

In the adaptive mode the noise reduction coefficient k is effected by the lower frequencies of the difference (new pixel - old pixel, N - O) of the luminance signal. The effect of the k-factor can be set by using a look-up table (LUT). Fig. 87 gives a block diagram of the DNR circuit.

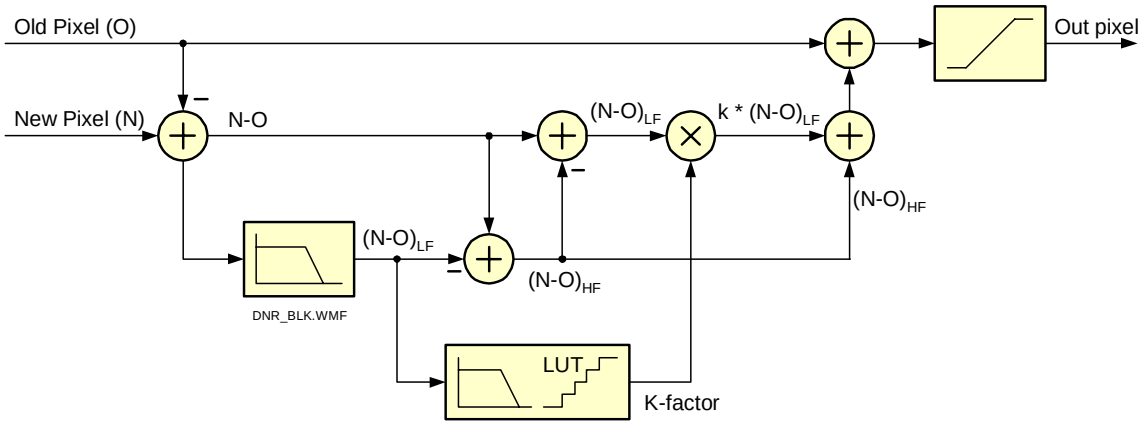


Fig. 87 DNR block diagram

The new pixel ‘out_pixel’ which will be stored in the memory is calculated by the addition of the previous signal (last field, ‘Old_pixel’) and a signal dependent part of the new input signal ‘New_pixel’. The difference between the old and the new input signal ‘N – O’ is low-pass filtered. This signal ‘$(N – O)_{LF}$’ is used to determine the proper k-factor and is subtracted from the difference signal from the input in order to obtain the high frequency part ‘$(N – O)_{HF}$’.

If the difference between the input signals is low, only a small share of the new signal is used. If the difference is high, e.g. in case of movements or vertical contrast, the new signal will become a higher share of the output signal.

In order to get an optimum k-factor which actually indicates the measure of noise reduction for each pixel the signal ‘$(N – O)_{LF}$’ is low-passed once more and the absolute value is calculated. By means of a filter the average is found and compared with the LUT in order to determine the k-factor.

This factor is used to assess the low-passed signal ‘$(N – O)_{LF}$’ which will be added to the old pixel. The high-pass part of the signal is added by using a fixed factor as well.

The first low-pass filter can be bypassed. In this case the high-pass term will become zero and the low frequent spectrum term must be interpreted as full frequency range.

The noise reduction coefficient (k-factor) calculated from of the luminance signal can optionally be coupled to the color processing circuit in order to control the chrominance noise reduction. The advantage of coupling is that cross color is reduced. The disadvantage is possible smearing of moving colored objects that have little Y-contrast with the background. Therefore, it is suggested to use coupling in applications without active comb-filter and no coupling whenever a comb filter is activated.

Further local adaptability is possible by using (partial) vector compensation. Then the horizontal component of the estimated motion vector is used to shift moving objects from the previous frame/field in the direction of the object in the current field. It can be set to 1/8 ... 7/8 of the estimated vector. Suggested is a vector compensation value of 6/8. A positive effect of vector compensation is the motion that is brought into the ‘dirty window effect’.

In order to be able to judge the effect of the noise reduction a split-screen mode can be used. In this mode the left side of the screen is set to a fixed k while the right side runs in adaptive k mode.

5.8 DNR in the SAA4979 and SAA4998

On the MK14-EM modules two noise reduction functions are available. They are basically identical. Both support 'noise shaping', a feature which eliminates left over image artefacts at sudden scene changes. The noise reduction loop in the SAA4979 is field based, the one in the SAA4998 is frame based. Therefore this one has advantages in progressive scan modes.

6. Frontend and Color Decoder

The module accepts two separate input signals for display on the screen, either side-by-side in a double window mode or with the second signal being inserted in the main picture (PIP, picture-in-picture). Each input signal is supplied to a color decoder of the SAA7118 type and can be CVBS, YC, or RGB. The output of each decoder is in digital format and is supplied to the scan conversion unit.

Besides analog signals the module accepts a digital input signal in ITU-656 format. It can be input into the extension port of the second color decoder or directly fed into the scan conversion unit.

6.1 The SAA7118

The SAA7118 is a digital multistandard color decoder with an adaptive comb filter and integrated input selector. The datasheet gives extensive information about the IC, therefore only a rough overview of the device and its features will be given here. A table lists the data sent to the device by I²C bus during initialization.

6.2 Functional blocks

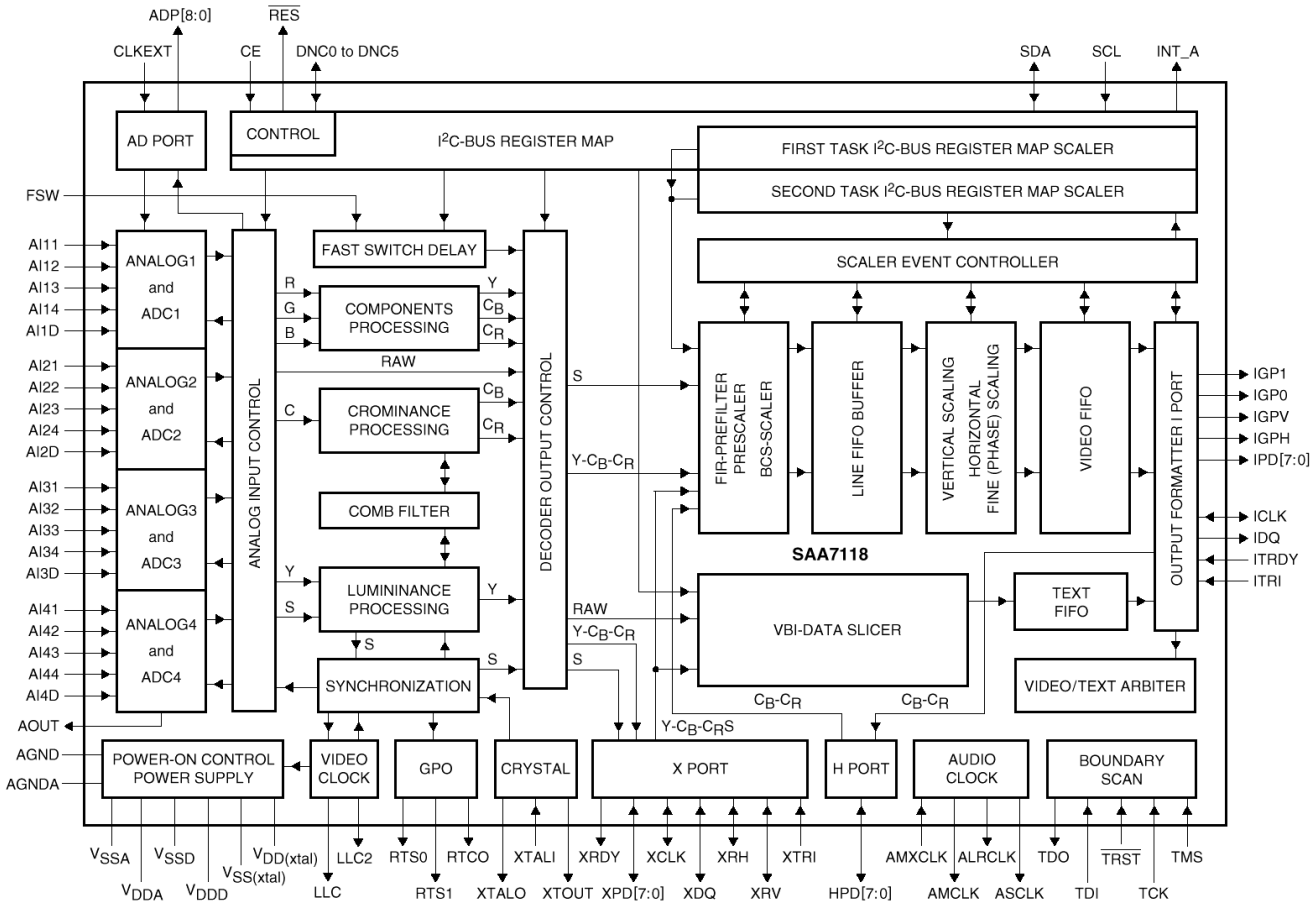


Fig. 88 Block diagram of the SAA7118

Video acquisition

There are four input channels, each one equipped with a clamping circuit, amplifier, antialiasing filter and a 9-bit ADC. Each channel can be connected to one of four input pins. So up to sixteen analog CVBS or eight analog

Y + C or four analog component inputs (RGB or YUV) can be connected. A static gain can be programmed for each channel, or Automatic Gain Control (AGC) can be selected.

Video decoder

The video decoder has a digital PLL and accepts all standard (PAL, NTSC, SECAM) and non-standard video sources (VTR). An adaptive 2/4-line comb filter is used for chrominance/luminance separation and enables increased luminance and chrominance bandwidth and reduces cross color and cross luminance. Brightness, contrast and saturation is adjusted separately for composite and component signals.

Component video processing

RGB as well as $Y\text{-}P_B\text{-}P_R$ (YUV) component input signals are processed with fast blanking being supported between CVBS and synchronous component inputs.

Video scaler

The scaler permits horizontal and vertical downscaling and upscaling from a zoom factor of approx. 1.18 (or 2.36 if the additional H output port is used) down to 1/64. The signal for the scaler can be input either from the decoder or from the expansion port (X-port). The scaler output data can be controlled in brightness, contrast and saturation.

Vertical blanking interval (VBI) data decoder and slicer

The SAA7118 contains a versatile VBI-data decoder generating teletext data, close caption, wide screen signalling etc. The extracted bytes are inserted into the digital output data of the I-port.

Audio clock generation

An audio clock is generated which is locked to the video field frequency. This ensures synchronous playback of audio and video after digital recording or non-linear editing.

Digital I/O interfaces

Real-time control and status information is output by the real-time port RTCO, RTS1 and RTSO. Various real-time status information can be selected for the RTS pins.

The digital video expansion port (X-port) outputs unscaled digital decoder data or inputs digital data for the scaler.

The digital image port (I-port) outputs scaled video data or data from the VBI data decoder.

The digital host port (H-port) can be used to expand the image port or the expansion port from 8 to 16 bits.

6.3 Initialisation data for the SAA7118

The following table lists a sample data set which is sent to the device for initialization by the Philips demonstration software for the MK14 module. After the data are sent to the device, a software reset must be performed. This is done setting bit SWRST (88H, bit 5) to logic 0.

Subaddr. (hex)	Value (hex)	Bit(s)	Function
Front end			
01	07	-0-- ---- --00 ---- ---- 0111	white peak control = active update hysteresis for 9-bit gain = off increment horizontal delay = 7
02	C0	11-- ---- --00 0000	analog function: amplifier plus anti-alias filter active mode 0: CVBS (automatic gain) from AI11
03	10	-0-- ---- --0- ---- ---1 ---- ---- 0--- ---- -0-- ---- --0- ---- ---0	normal clamping if decoder is in unlocked state short vertical blanking (AGC disabled during equalization and serration pulses) colour peak control = off automatic gain control = active automatic gain controlled by MODE5 to MODE0 03H[1] and 05H[7:0]: static gain control channel 2 = 90H 03H[0] and 04H[7:0]: static gain control channel 1 = 90H
04	90	1001 0000	03H[0] and 04H[7:0]: static gain control channel 1 = 90H
05	90	1001 0000	03H[1] and 05H[7:0]: static gain control channel 2 = 90H
Decoder			
06	50	0101 0000	horizontal sync start = 80 * 8 LLC
07	F4	1111 0100	horizontal sync stop = -12 * 8 LLC
08	98	1--- ---- -0-- ---- --0- ---- ---1 1--- ---- -0-- ---- --00	automatic field detection = on field selection = 50 Hz, 625 lines ODD/EVEN signal toggles only with interlaced sources horizontal time constant = fast locking mode horizontal PLL = closed vertical noise reduction = normal mode
09	42	0--- ---- -1-- ---- --0- ---- ---0 ---- ---- 0010	chrominance trap or luminance comb filter = active adaptive luminance comb filter = active processing delay in non-comb filter mode is equal to internal pipelining delay remodulation bandwidth for luminance = small (narrow chroma notch ⇒ higher luminance bandwidth) luminance resolution enhancement filter 6.8 dB at 4.1 MHz
0A	80	1000 0000	luminance brightness control: offset = ITU level
0B	44	0100 0100	luminance contrast control: gain = 1.063 (ITU level)
0C	40	0100 0000	chrominance saturation control: gain = 1.0 (ITU level)

Subaddr. (hex)	Value (hex)	Bit(s)		Function
0D	00	0000	0000	chrominance hue control: hue phase = 0 deg.
0E	83	1---	----	clear DTO
		-000	----	color standard selection (preferred) = PAL BGDHI (4.43 MHz) or NTSC M (3.58 MHz)
		----	0---	chrominance vertical filter and PAL phase error correction = on
		----	-0--	color time constant = nominal
		----	--1-	14[2] and 0E[1]: 01 = automatic chrominance standard detection = active, filter settings and sharpness control are preset to default values according to the detected standard and mode
		----	---1	adaptive chrominance comb filter = active
0F	2D	0---	----	automatic chrominance gain control = on
		-010	1101	chrominance gain value
10	01	00--	----	fine offset adjustment B - Y component = 0 LSB
		--00	----	fine offset adjustment R - Y component = 0 LSB
		----	0---	chrominance bandwidth = small
		----	-001	small chrominance bandwidth / large luminance bandwidth
11	00	0---	----	automatic color killer enabled
		-0--	----	polarity of RTS1 output signal = non-inverted
		--00	----	fine position of horizontal sync = 0
		----	0---	polarity of RTS0 output signal = non-inverted
		----	-000	luminance delay compensation = 0
12	00	0000	----	RTS0 output = 3-state
		----	0000	RTS1 output = 3-state
13	80	1---	----	RTCO output enabled
		-0--	----	X-port XRH output = HREF
		--00	----	X-port XRV output = V123
		----	0---	horizontal lock indicator = copy of inverted HLCK status bit
		----	-000	XPD7..XPD0 output format = ITU 656
14	00	0---	----	compatibility bit for SAA7199 = off
		-0--	----	update time interval for AGC value = horizontal update (once per line)
		--00	----	23H[7] and 14H[5:4]: analog test select = AOUT connected to ground
		----	0---	XTOUT output = 3-stated
		----	-0--	14H[2] and 0EH[1]: 00 = automatic chrominance standard detection disabled
		----	--00	ADC sample clock phase delay = application dependent
15	20	0010	0000	17H[0] and 15H[7:0]: start of VGATE pulse and polarity change of FID pulse = 32
16	FE	1111	1110	17[1] and 16[7:0]: stop of VGATE pulse = 254

Subaddr. (hex)	Value (hex)	Bit(s)	Function
17	18	0--- ---- -0-- ---- --01 1--- ---- -0-- ---- --0- ---- ---0	LLC output = enabled LLC2 output = enabled standard detection search loop latency = three fields VGATE position according to subaddr. 15H and 16H MSB of VGATE stop (subaddr. 16H) MSB of VGATE start (subaddr. 15H)
18	42	0100 0010	raw data gain control
19	80	1000 0000	raw data offset control
Component processing and interrupt masking			
23	00	0--- ---- -0-- ---- --0- ---- ---0 ---- ---- -0-- ---- --0- ---- ---0	23H[7] and 14H[5:4]: analog test select = AOUT connected to ground AD port is set to 3-state all ADCs are clocked by the internal generated line-locked clock clamping is dependent on HLNRS (03H[6]) external source switch indicator input disabled 23H[1] and 25H[7:0]: static gain control channel 4 = 90H 23H[0] and 24H[7:0]: static gain control channel 3 = 90H
24	90	1001 0000	23H[0] and 24H[7:0]: static gain control channel 3 = 90H
25	90	1001 0000	23H[1] and 25H[7:0]: static gain control channel 4 = 90H
29	D0	1--- ---- -1-- ---- --01 ---- ---- 0--- ---- -000	fast switch enable = pixelwise switching between decoded CVBS signal and component input signal fast switch input polarity: FSW = 1: decoded CVBS signal, FSW = 0: component signal fast switch input delay adjustment relative to component input signal = +1 pixel component luminance peaking = disabled component input delay adjustment relative to decoded CVBS signal = 0 pixel
2A	80	1000 0000	luminance brightness control component part
2B	44	0100 0100	luminance contrast control component part
2C	40	0100 0100	chrominance saturation control component part
2D	00	**Interrupt mask 1:** ---0 ---- ---- 0--- ---- -0-- ---- ---0	interrupt 'VPS signal detected/lost' = disabled (corresponding flag: 60H[4]) interrupt 'PALplus detected/lost' = disabled (corresponding flag: 60H[3]) interrupt 'closed caption detected/lost' = disabled (corresponding flag: 60H[2]) interrupt 'error output formatter' = disabled (corresponding flag: 8FH[2])

Subaddr. (hex)	Value (hex)	Bit(s)	Function
2E	00		**Interrupt mask 2:**
		-0-- ----	interrupt 'horizontal PLL locked/unlocked' = disabled (corresponding flag: 1EH[6])
		---- --0-	interrupt 'colour standard changed 1' = disabled (corresponding flag: 1EH[1])
		---- ---0	interrupt 'colour standard changed 0' = disabled (corresponding flag: 1EH[0])
2F	00		**Interrupt mask 3:**
		0--- ----	interrupt 'interlaced/non-interlaced source' = disabled (corresponding flag: 1FH[7])
		-0-- ----	interrupt 'horizontal and vertical lock reached/lost' = disabled (corresponding flag: 1FH[6])
		--0- ----	interrupt 'field frequency has changed' = disabled (corresponding flag: 1FH[5])
		---- 0---	interrupt 'colour stripe type 3 burst detected/lost' = disabled (corresponding flag: 1FH[3])
		---- -0--	interrupt 'colour stripe burst (any type) detected/lost' = disabled (corresponding flag: 1FH[2])
		---- --0-	interrupt 'copy protected signal found/lost' = disabled (corresponding flag: 1FH[1])
		---- ---0	interrupt 'ready for capture/not ready' = disabled (corresponding flag: 1FH[0])
Audio clock generation			
30	BC		audio master clock cycles per field
31	DF		
32	02		
34	CD		audio master clock nominal increment
35	CC		
36	3A		
38	03	0000 0011	clock ratio audio master clock to serial bit clock
39	20	0010 0000	clock ratio serial bit clock to channel select clock
Data slicer and data type control			
40	00	-0-- ----	Hamming check for 2 bytes after framing code
		--0- ----	one frame code error allowed
		---0 ----	amplitude searching active
41..56	00	0000 0000	VBI-data slicer line control registers LCR2 to LCR23 = teletext EuroWST, CCST
57	FF	1111 1111	VBI-data slicer line control registers LCR24 = video component signal, active video region
58	00	0000 0000	framing code for programmable data types

Subaddr. (hex)	Value (hex)	Bit(s)	Function
59	47	0100 0111	5BH[2:0] and 59H[7:0]: horizontal offset for slicer = 347H
5A	09	0000 1001	5BH[4] and 5AH[7:0]: vertical offset for slicer = 9H
5B	83	1--- ---- -0-- ---- ---0 ---- ---- -011	field offset: invert field indicator recoding: leave data unchanged 5BH[4] and 5AH[7:0]: vertical offset for slicer = 9H horizontal offset for slicer, MSBs
5D	80	1--- ---- --00 0000	F and V output is taken from decoder real-time signals EVEN_ITU and VBLNK_ITU ANC header framing
5E	00	--00 0000	sliced data identification code
Scaler and interface global settings			
80	50	-1-- ---- --0- ---- ---1 ---- ---- 0--- ---- -0-- ---- --00	task of register set A is enabled task of register set B is disabled VBI-data slicer defines the F and V timing of the scaler output IDQ generation only for valid data IDQ pin carries data qualifier ICLK output and back-end clock is line-locked clock LLC from decoder
83	21	--10 ---- ---- -0-- ---- --01	XCLK phase shifted by approximately 3 ns XRDY output signal is A/B task flag from event handler (A = 1) X-port output is enabled by software
84	00	00-- ---- --00 ---- ---- 00-- ---- --00	86H[5] and 84H[7:6]: IGP0 is output field ID, as definded by OFIDC (90H[6]) 86H[4] and 84H[5:4]: IGP1 is output field ID, as defined by OFIDC (90H[6]) IGPV is a V-gate signal, framing scaled output lines IGPH is a H-gate signal, framing the scaler output
85	10	10-- ---- --0- ---- ---0 ---- ---- 0--- ---- -0-- ---- --0- ---- ---0	X-port Dword bytes swapped: D2 D3 D0 D1 ⇒ 00 SAV FF 00 C_R 0 Y1 C_B 0 Y0 X-port video data limited to range 1 to 254 I-port reference signal IGP0 at default polarity I-port reference signal IGP1 at default polarity I-port reference signal IGPV at default polarity (1 = active) I-port reference signal IGPH at default polarity (1 = active) I-port reference signal IDQ at default polarity (1 = active)
86	45	01-- ---- --0- ---- ---0 ---- ---- 01-- ---- --01	I-port signal definition: only video data is transferred see subaddr. 84 see subaddr. 84 I-port FIFO flag almost full level ≥ 24 Dwords I-port FIFO flag almost empty level < 8 Dwords
87	20	00-- ---- --10 ---- ---- --00	IDQ = gated clock default output phase ICLK phase shifted by approximately 3 ns I-port output is disabled by software

Subaddr. (hex)	Value (hex)	Bit(s)	Function
88	30	00-- ---- --1- ---- ---1 ---- ---- -0-- ---- --0- ---- ---0	digitized ADC1 signal is fed to port ADP[8:0] scaler is switched back to operation DPROG = 1 can be used to assign that the device has been programmed audio clock generation active scaler is in operational mode decoder and VBI slicer are in operational mode
Scaler task A: scaler input configuration and output format settings			
90	00	0--- ---- -0-- ---- --00 0--- ---- -0-- ---- --00	scaler SAV/EAV byte bit D7 and task flag = 1, default output field ID is field ID from scaler input active task is carried out directly if active task is finished, handling is taken over by the next task event handler triggers immediately after finishing a task
91	08	0--- ---- -0-- ---- --00 ---- ---- 1--- ---- -00- ---- ---0	SAV/EAV code bit D5 (V) and V-gate on pin IGPV as generated by the internal processing SAV/EAV code bits D6 and D5 (F and V) may change between SAV and EAV scaler input source is data from decoder scaler input source is a continuous data stream, which cannot be interrupted chroma is provided every line scaler input format is Y-CB-CR 4 :2 :2 like sampling scheme
92	42	X-port input reference signal definitions: 0--- ---- -1-- ---- --0- ---- ---0 ---- ---- 0--- ---- -0-- ---- --1- ---- ---0	reference edge for field detection is falling edge of XRV field ID (decoder and X-port field ID) is inverted XRV is a V-sync or V-gate signal rising edge of XRV input and decoder V123 is vertical reference reference signals are taken from XRH and XRV rising edge of XRH input is horizontal reference data are qualified at XDQ input at logic 0 XCLK input clock and XDQ input qualifier are needed
93	80	I-port output format and configuration: 1--- ---- -0-- ---- --0- ---- ---0 0--- ---- -000	ITU 656 like SAV/EAV codes are inserted in the output data stream, framed by a qualifier Dwords are transferred byte wise, see subaddress 85H all lines will be output no leading Y only line, before 1st Y + C_B-C_R line is output 4 :2 :2 Dword formatting
Scaler task A: input and output window definition			
94	00		horizontal input acquisition window offset
95	00		
96	D0		horizontal input acquisition window length ($2D0_H$ = 720_D)
97	02		

Subaddr. (hex)	Value (hex)	Bit(s)	Function
98	00		vertical input acquisition window offset (0)
99	00		
9A	71		vertical input acquisition window length ($271_H = 625_D$)
9B	02		
9C	D0		horizontal output window length ($2D0_H = 720_D$)
9D	02		
9E	71		vertical output window length ($271_H = 625_D$)
9F	02		
Scaler task A: prefiltering and prescaling			
A0	01	--00 0001	horizontal integer prescaling ratio = 1
A1	00	--00 0000	horizontal prescaler accumulation sequence length = 1
A2	00	00-- ---- --00 ---- ---- 0--- ---- -000	chrominance FIR filter bypassed luminance FIR filter bypassed prescaler DC gain: weighting of all accumulated samples is factor '1' prescaler output is renormalized by gain factor = 1
A4	80	1000 0000	luminance brightness control
A5	50	0101 0000	luminance contrast control
A6	40	0100 0000	chrominance saturation control
Scaler task A: horizontal phase scaling			
A8	00		horizontal luminance scaling increment (400_H = scale 1)
A9	04		
AA	00	0000 0000	horizontal luminance phase offset = 0
AC	00		horizontal chrominance scaling increment (200_H = scale 1, 1/2 of horizontal luminance scaling increment)
AD	02		
AE	00	0000 0000	horizontal chrominance phase offset = 0
Scaler task A: vertical scaling			
B0	00		vertical luminance scaling increment = 1
B1	04		
B2	00		vertical chrominance scaling increment = 1
B3	04		

Subaddr. (hex)	Value (hex)	Bit(s)	Function
B4	01	---0 ---- ---- ---1	vertical scaling mode: no mirroring vertical scaling performs higher order accumulating interpolation, better alias suppression
B8	00		vertical chrominance phase offset = 0
B9	00		
BA	00		
BB	00		
BC	00		vertical luminance phase offset = 0
BD	00		
BE	00		
BF	00		
Scaler task B: scaler input configuration and output format settings			
C0	00	0--- ---- -0-- ---- --00 0--- ---- -0-- ---- --00	scaler SAV/EAV byte bit D7 and task flag = 1, default output field ID is field ID from scaler input active task is carried out directly if active task is finished, handling is taken over by the next task event handler triggers immediately after finishing a task
C1	08	0--- ---- -0-- ---- --00 ---- ---- 1--- ---- -00- ---- ---0	SAV/EAV code bit D5 (V) and V-gate on pin IGPV as generated by the internal processing SAV/EAV code bits D6 and D5 (F and V) may change between SAV and EAV scaler input source is data from decoder scaler input source is a continuous data stream, which cannot be interrupted chroma is provided every line scaler input format is Y-CB-CR 4 :2 :2 like sampling scheme
C2	42	0--- ---- -1-- ---- --0- ---- ---0 ---- ---- 0--- ---- -0-- ---- --1- ---- ---0	X-port input reference signal definitions: reference edge for field detection is falling edge of XRV field ID (decoder and X-port field ID) is inverted XRV is a V-sync or V-gate signal rising edge of XRV input and decoder V123 is vertical reference reference signals are taken from XRH and XRV rising edge of XRH input is horizontal reference data are qualified at XDQ input at logic 0 XCLK input clock and XDQ input qualifier are needed
C3	00	0--- ---- -0-- ---- --0- ---- ---0 0--- ---- -000	I-port output format and configuration: no ITU 656 like SAV/EAV codes are available Dwords are transferred byte wise, see subaddress 85H all lines will be output no leading Y only line, before 1st Y + C_B -C_R line is output 4 :2 :2 Dword formatting

Subaddr. (hex)	Value (hex)	Bit(s)	Function
Scaler task B: Input and output window definition			
C4	0A		horizontal input acquisition window offset
C5	00		
C6	C0		horizontal input acquisition window length ($2C0_H = 704_D$)
C7	02		
C8	16		vertical input acquisition window offset ($16_H = 22_D$)
C9	00		
CA	18		vertical input acquisition window length ($118_H = 118_D$)
CB	01		
CC	60		horizontal output window length ($160_H = 352_D$)
CD	01		
CE	8A		vertical output window length ($8A_H = 138_D$)
CF	00		
Scaler task B: Prefiltering and prescaling			
D0	01	--00 0001	horizontal integer prescaling ratio = 1
D1	01	--00 0000	horizontal prescaler accumulation sequence length = 2
D2	A1	01-- ---- --01 ---- ---- 0--- ---- -001	FIR prefilter control: H_uv(z) = 1/4 (1 2 1) FIR prefilter control: H_y(z) = 1/4 (1 2 1) prescaler DC gain: weighting of all accumulated samples is factor ‘1’ prescaler output is renormalized by gain factor = 1/2
D4	80	1000 0000	luminance brightness control
D5	20	0010 0000	luminance contrast control
D6	20	0010 0000	chrominance saturation control
Scaler task B: horizontal phase scaling			
D8	00		horizontal luminance scaling increment
D9	08		
DA	00	0000 0000	horizontal luminance phase offset = 0
DC	00		horizontal chrominance scaling increment
DD	04		
DE	00	0000 0000	horizontal chrominance phase offset = 0
Scaler task B: vertical scaling			

Subaddr. (hex)	Value (hex)	Bit(s)	Function
E0	00		vertical luminance scaling increment
E1	08		
E2	00		vertical chrominance scaling increment
E3	08		
E4	01	---0 ---- ---- ---1	vertical scaling mode: no mirroring vertical scaling performs higher order accumulating interpolation, better alias suppression
E8	00		vertical chrominance phase offset
E9	00		
EA	30		
EB	30		
EC	00		vertical luminance phase offset
ED	00		
EE	30		
EF	30		

7. Available Hardware

7.1 The IPQ module MK14-EM

Fig. 89 gives the block diagram of the MK14-EM module and fig. 90 shows a picture of it. The two color decoders each have CVBS, Y/C and RGB inputs which can be selected by I²C bus. Besides these analog inputs a digital YUV interface in ITU-656 format is available. Digital signals can be input as main channel signal into the SAA4979 or as subchannel signal via the second color decoder.

Data transfer from both color decoders to the SAA4979 is in digital format. The main channel decoder interfaces directly to the SAA4979, the subchannel via a buffer memory. One reason for this is that data must be synchronized to the main channel data for display on the screen, and the second reason is that a buffer memory is necessary for downscaling the picture in case of PIP (picture-in-picture). Writing to this memory is controlled by the color decoder and reading is under control of the SAA4979. The buffer memory is inside the SAA4998.

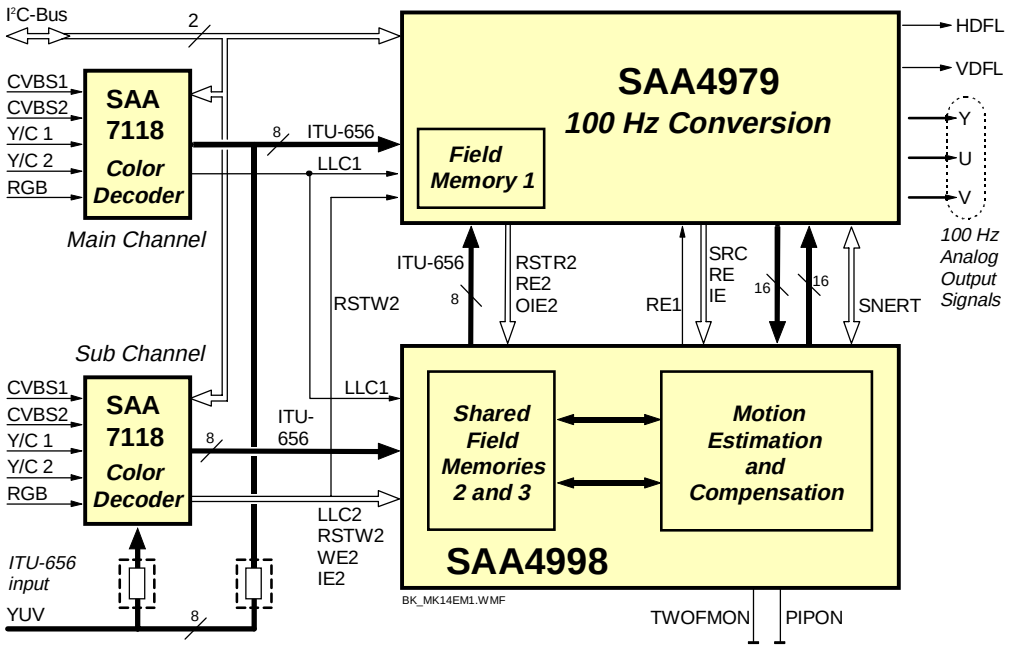


Fig. 89 Block diagram of the MK14-EM module

Provision have been made on the board to accept also external ITU-656 data. If they are to be displayed in the main channel then the resistors R530..R538 have to be placed on the board. To avoid data collision on the bus the outputs of the main color decoder need to be disabled[1]. For displaying external ITU-656 data on the subchannel they have to be input at the X-port of the subchannel decoder, and resistors R539..R547 have to be placed on the board.[2]

The two configuration pins of the SAA4998 *TwoFMon* and *PIPon* are set to logic 0, so the set starts up in FALCONIC mode after reset. That means that both internal memories are used for motion compensation. By software PIP can be activated and one or both memories be allocated as PIP buffer.

1. see reg. 87_H of the SAA7118 for I-port output enable/disable
2. for data input at the X-port of the SAA7118 select:
   I-port and scaler back-end clock selection: reg. $80_H = x1_H$
   X-port I/O enable and output clock phase control: reg. $83_H = 10_H$
   X-port formats and configuration: reg. $91_H/C1_H = 18_H$
   X-port input reference signal definitions: reg. $92_H/C2_H = 19_H$

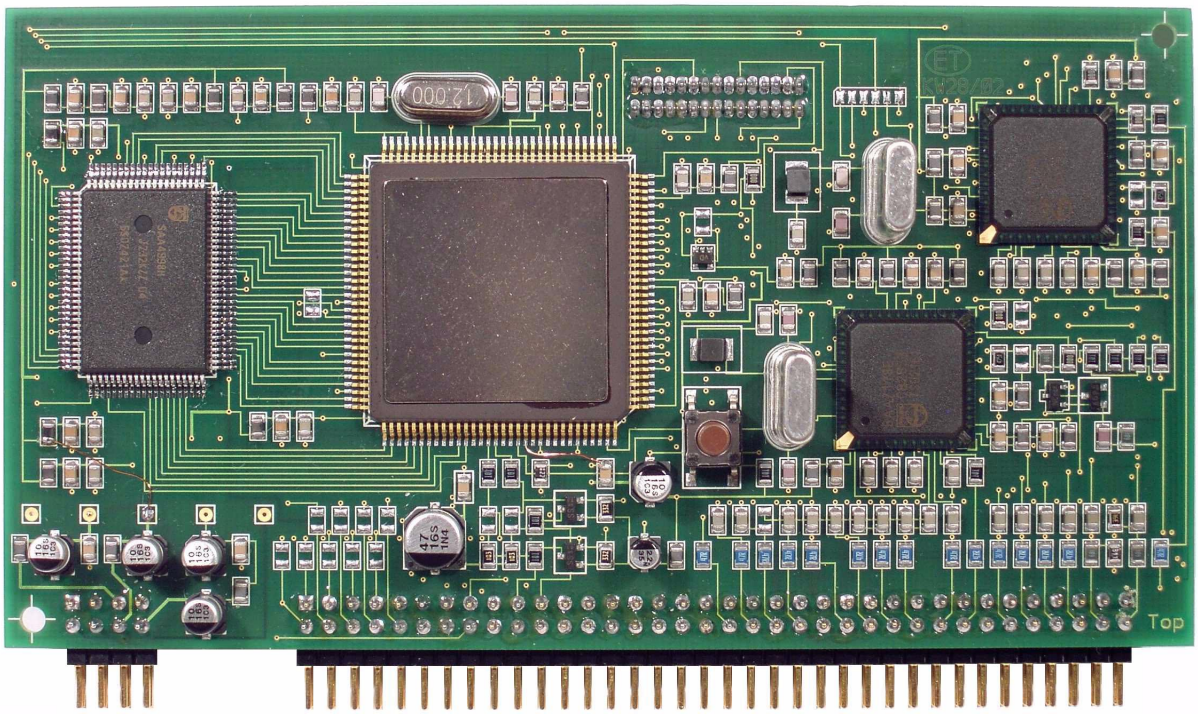


Fig. 90 Top view of the MK14-EM board

The board can be run without motion compensation and PIP. In this case no subchannel color decoder SAA7118 and no SAA4998 need to be assembled. The read enable output signal RE is to be connected to the input signal REI by putting in place the resistor R204 (see fig. 91).

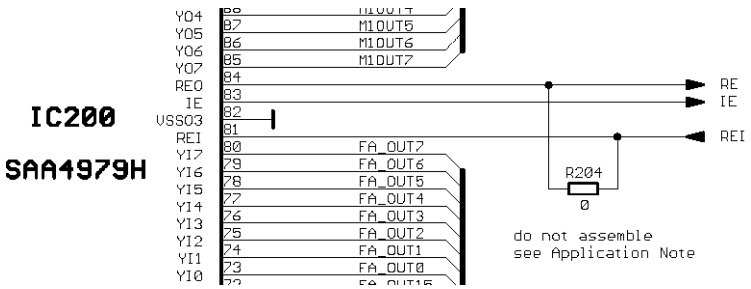


Fig. 91 Running the MK14-EM board without subchannel and motion compensation.

For development purposes there is a special version of the SAA4979 available. It has pins to which an external EPROM or software emulator can be connected. Running such development samples requires that an external EPROM piggy-back board is connected to CON200 (on the rear side of the board). The final version of the SAA4979 runs on its internal software and cannot use any external ROM, so neither this connector nor the EPROM socket is needed. For the development samples port pin P1.2 defines whether the internal or external software is used. Connecting P1.2 to GND (putting R205 in place) means external ROM mode, without R205 P1.2 will have HIGH level (due to its internal pull-up) and will define internal ROM mode.

The digital decoders SAA7118 can run on either one of two I²C addresses: 40_H or 42_H. The Philips I²C control software expects the main channel decoder at address 42_H and the subchannel decoder at address 40_H. So for

the main channel R101 is put in place (R102 is not), this generates a LOW level at pin RTCO (= address 42_H), for the sub channel R705 is assembled (R706 is not) generating a HIGH level at pin RTCO (= address 40_H).

On the board each color decoder has its own clock generation based on a 24.576 MHz crystal. However, it is also possible to use only the clock generator of the main channel decoder and feed its clock from the output pin XTOUT via R703 to the clock input pin XTALI of the sub channel decoder. In this case components X700, L700, C722, C723 and C724 are not assembled.

When inputting RGB signals into the main channel SAA7118 decoder it has to be decided whether synchronization signals are to be taken from the GIN-input (green) or if they are supplied separately. In case of RGB input mode the SAA7118 takes A/D converter no. 1 (pin AI13) for sync processing. Therefore either C124 must be assembled (synchronization signals taken from GIN) or C106 (synchronization signals supplied separately at input CVBS1).

7.2 Test and evaluation environment

The modules MK14-EM is plugged onto a mother board for operation. This mother board offers a standard 21-pin SCART socket for RGB and CVBS signals, two BNC and two cinch sockets for CVBS, and a hosiden socket for Y/C. A small matrix permits to configure which mother board input signal is connected to which IPQ board input pin. If RGB signals are used, the CVBS signal of the SCART plug is needed for synchronization and has to be switched to the CVBS1 pin of the IPQ board, see also fig. 93.

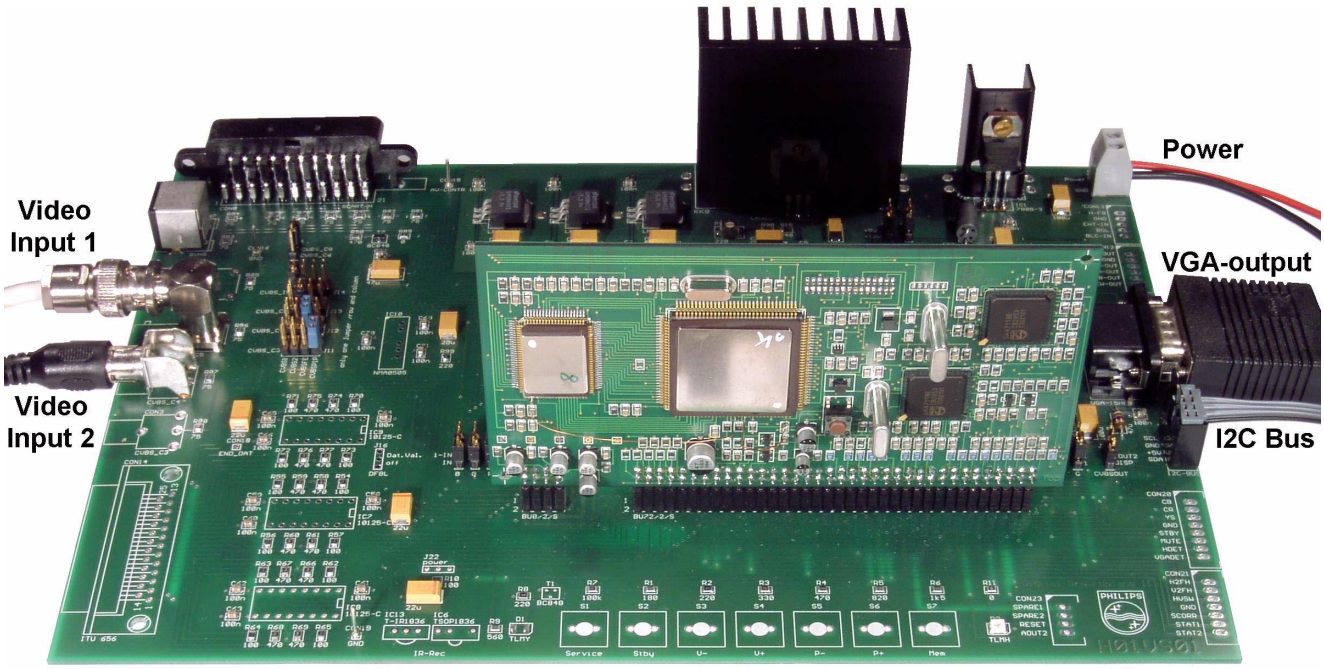


Fig. 92 IPQ board MK14-EM on mother board

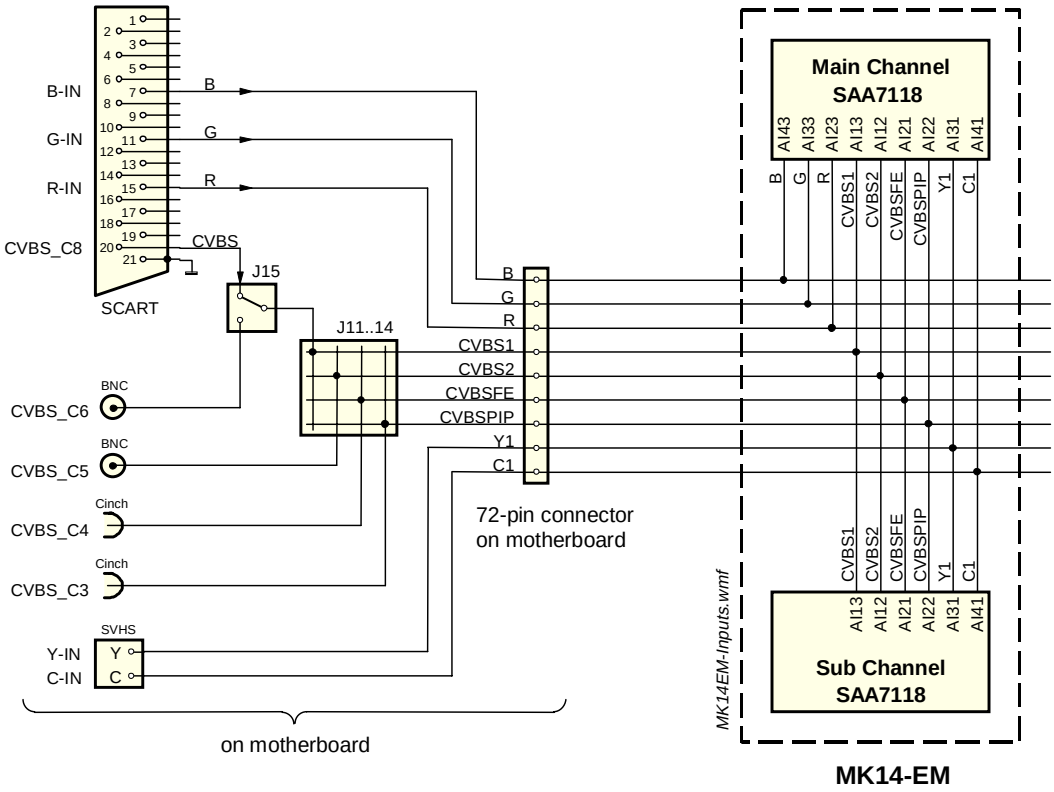


Fig. 93 Input selection on mother board

8. Application environment

8.1 Motion compensation in a TV set

The IPQ module MK14-EM is intended to be used for scan conversion purposes in the TV environment. With the two color decoders SAA7118 a digital front end is on board delivering two ITU data streams to the scan conversion IC SAA4979. ITU1 is the main channel data stream, ITU2 is the subchannel delivering data for the second (simultaneous) picture on the screen, e. g. side by side in a double window mode or as a small picture within the main one (PIP - picture in picture).

Together with the data on ITU1 the main channel decoder also delivers the main input clock LLC1 (Line Locked Clock, 27 MHz) which serves as reference for the SAA4979's main clock. The subchannel decoder is independent from the main channel one and writes its data into a buffer memory inside the SAA4998 using its LLC2 clock. In order to display subchannel data together with main channel data, the ITU2 data must be synchronized horizontally and vertically to the ITU1 data. For this the SAA4979 generates the memory control signals RSTR2, RE2 and OIE2 and reads the data from the buffer memory using clock LLC1. RSTR2 resets the read address pointer in the buffer memories, i. e. it defines the start of reading (upper left corner of the active picture). By activating RE2 the SAA4979 fetches picture data line by line. OIE2 defines from which one of the two buffer memories data is read.

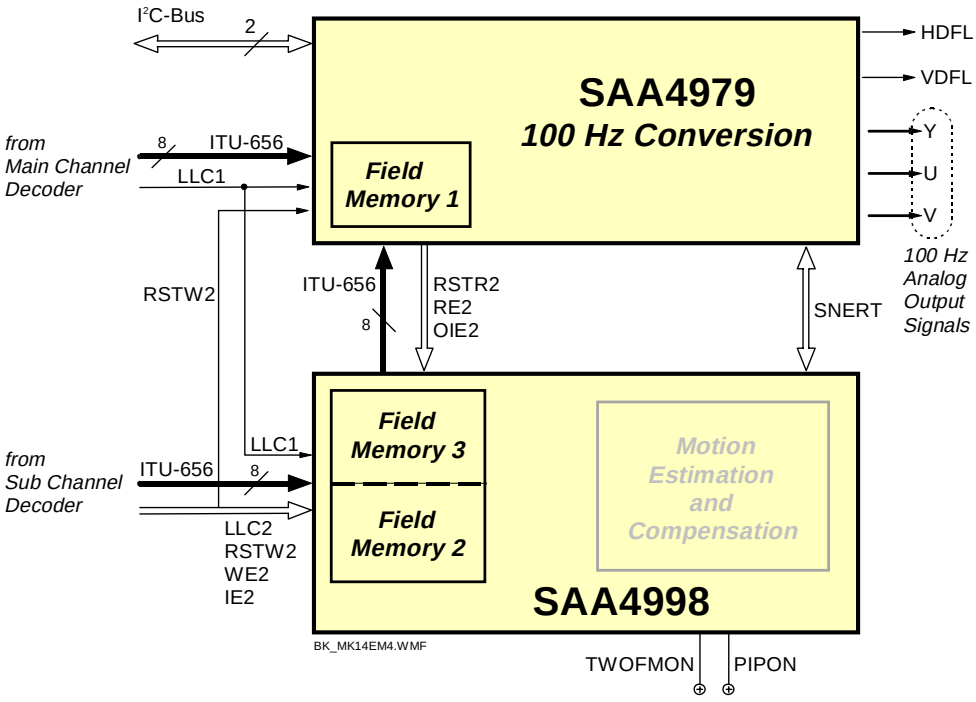


Fig. 94 Data flow in two-channel display mode

In order to store ITU data only 8 of the 12 bits of the memory are used. However since luminance and chrominance data are transmitted alternatingly by the decoder at a clock rate of 27 MHz, two bytes are needed for each pixel. With its storage capacity of 245772 words one memory cannot hold a complete field any more. This however is not necessary since the subchannel always deals with scaled-down pictures: the largest picture with 50% of the original size occurs in double window mode, in PIP mode the amount of picture data is even smaller.

There are two memories used to buffer the subchannel data. This is done in order to minimize effects due to false motion sequence or false raster position in the subchannel display. A momentary motion disturbance occurs whenever the write and read pointers pass each other in whatever direction: when the read pointer passes

the write pointer one motion phase is shown twice, when the write pointer passes the read pointer then a motion phase is left out. False raster position means that the subchannel odd/even phase is different from the main channel odd/even phase, in this case a strong like flicker would occur in the subchannel display.

In case that both sources are of the same standard (both PAL or both NTSC) a passing of pointers occurs rather seldom, if the standards differ or if one source is from a VCR machine or the like, then this is more likely to happen. Therefore two main situations can be distinguished:

Both sources are of the same standard (PAL or NTSC) and write and read pointer speed is equal (double window mode):

In this case ("direct mode") any passing of pointer positions is not or only very seldom to be expected. The strategy here is to read the subchannel data from that memory which contains the correct field (odd or even) that fits to the main channel picture. If the read pointer passes the write pointer resulting in an odd/even error, then data is taken from the other memory.

Sources are of different standard or read pointer speed differs considerably from write pointer speed:

This is the case in any PIP (picture-in-picture) mode or e. g. if PAL and NTSC are to be displayed simultaneously in double-window mode. In this case the odd/even phase of the subchannel is determined and the raster corrected such that it fits with main channel. This correction is done by starting RE2 one line earlier or later.

8.2 Motion compensation in a DVD-player

DVDs usually contain movie material, i. e. scanned film pictures. These films are taken at a rate of 24 pictures per second. Whenever films are displayed on TV, artefacts in the presentation of moving objects occur which is due to the difference between movement rate (24 Hz) and the TV display rate (50 or 60 Hz).

Conversion in PAL mode (2:2 pull-down mode)
For conversion to TV (in PAL) the film is played back at 25 pictures per second (the speed increase of approx. 4% is negligible), and with each picture being scanned twice the TV field rate of 50 Hz is obtained. This is called 2-2 pull-down mode. Due to the field repetition a juddering of moving objects or a contouring along moving edges will be noticed, see chpt. 3.7 and fig. 7.

Conversion in NTSC mode (3:2 pull-down mode)
For conversion to the NTSC standard the film is played back at 24 pictures per second with consecutive pictures being scanned twice or three times alternately. This is called 3-2 pull-down mode see chpt. 3.7 and fig. 8. Besides a juddering similar to PAL conversion, here an additional low-frequency judder (12 Hz) can be noticed which is due to the changing repetition of fields (3 times and 2 times alternately).

In both modes movement artefacts can be compensated by using motion compensation ICs. Smooth movement representation without juddering is obtained. However the modes that the ICs offer is either field rate doubling (100 or 120 Hz field rate, 32 kHz line rate) or progressive scan (50 or 60 Hz field rate, 32 kHz line rate). Both modes require that a connected display unit (TV set, monitor, projector) accepts a horizontal line frequency of 32 kHz.

An attractive application would be to implement a movement compensation box directly in a DVD player. If the output were standard $1f_H$ format (50 Hz or 60 Hz), then any TV set (with 16 kHz deflection) could be used for display. Therefore an additional block is needed at the output of the motion compensation IC to change the progressive format to an interlaced format again. Basically a line memory (LM) is needed into which data is written at a clock frequency of 32 MHz and which is read at 16 MHz. Every second line is discarded. The vertical synchronization pulse needs to be adapted so interlaced scanning is performed by the display unit. A block diagram of such an application environment is given in fig. 95.

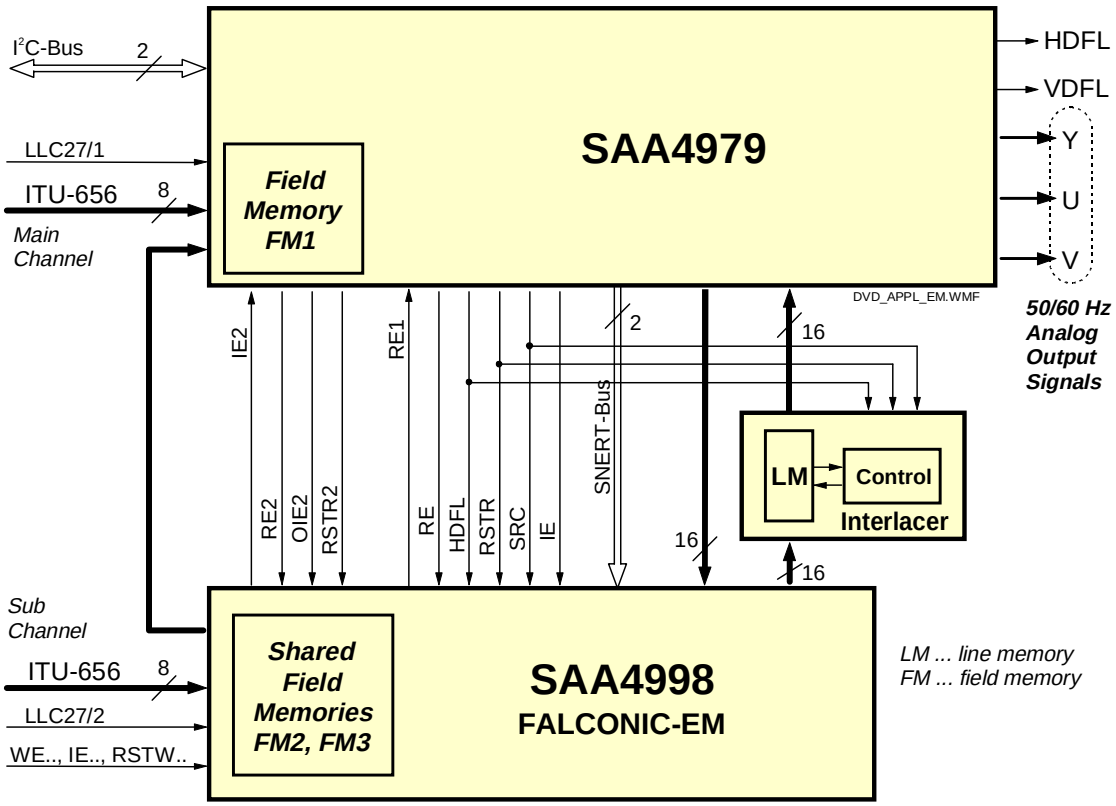


Fig. 95 Block diagram of motion compensation in a DVD player

8.3 Hints for a master control software

Tuner search mode

During tuner search mode usually OSD (on-screen display) is active. As long as no stable input signal is detected there is no stable synchronization signal for OSD. Therefore it is recommended to turn the SAA4979 into generator mode. In this mode the display PLL is open and a stable clock as well as stable H- and V-sync signals are generated.

The detection of a stable input signal is reflected in bit 6 (HLVLN) of register 1F in the SAA7118 color decoder. HLVLN = 1 indicates that both the horizontal and the vertical loop are unlocked and thus no stable input signal is detected.

Macrovision

Input signals which are copy-protected by Macrovision may create disturbances in the displayed signal if the automatic gain control (AGC) of the color decoder SAA7118 is active. Therefore gain control should be set to fixed by turning GAFIX (bit 2 in register 3) to 1. In this case user-controlled fixed gain settings are taken.

Macrovision signals are indicated by bit 1 in register 1F of the SAA7118: COPRO = 1.

9. PIP-Window construction using the PIP-Interface

9.1 General description

The definition of the PIP-Window builds up on the number of vertical pixels (equal to lines) and the number of horizontal pixels. The maximum number of pixels (vertical and horizontal) is limited by the actual active display state. The display runs on 32 MHz while the PIP is defined on the acquisition side which runs on 27 MHz. So the maximum display values converted to the acquisition side are:

	PAL (50 Hz field frequency)		NTSC (60 Hz field frequency)	
Direction	Non-interlaced	Interlaced	Non-interlaced	Interlaced
Horizontal	702	702	702	702
Vertical	570 (285 * 2)	569	466 (233 * 2)	473

702 pixels on the acquisition side result in 832 visible pixels on the display side.

The PIP-Interface function checks the size and position of the PIP-window. The table below shows the maximum values for the different modes.

	PAL (50 Hz field frequency)		NTSC (60 Hz field frequency)	
Direction	Non-interlaced	Interlaced	Non-interlaced	Interlaced
Horizontal	702	702	702	702
Vertical	575	575	483	483

If this size exceeds one or more of the maximum values, no PIP-Window will be created. In this case an error code is set so the master software can evaluate what went wrong. The following error codes are possible:

Error Code	Equivalent HEX-Code	Description
ERROR_TOP	04	Top (V_Start) Position is wrong
ERROR_LEFT	08	Left (H_Start) Position is wrong
ERROR_WIDTH	10	Right (H_Stop) Position is wrong
ERROR_LEFT_WIDTH	18	Left (H_Start) and Right (H_Stop) Positions are wrong
ERROR_HEIGHT	20	Bottom (V_Stop) Position is wrong
ERROR_TOP_HEIGHT	24	Top (V_Start) and Bottom (V_Stop) Positions are wrong

The definition of a PIP-window is shown in fig. 96.

If a frame around the PIP-window is defined, this frame builds up within the visible PIP-window. The outer lines of the PIP-frame are on the same lines / pixels as the PIP-window. So the visible PIP-window size is reduced by the width of the PIP-frame. The PIP-frame width and height are automatically limited to a maximum value of

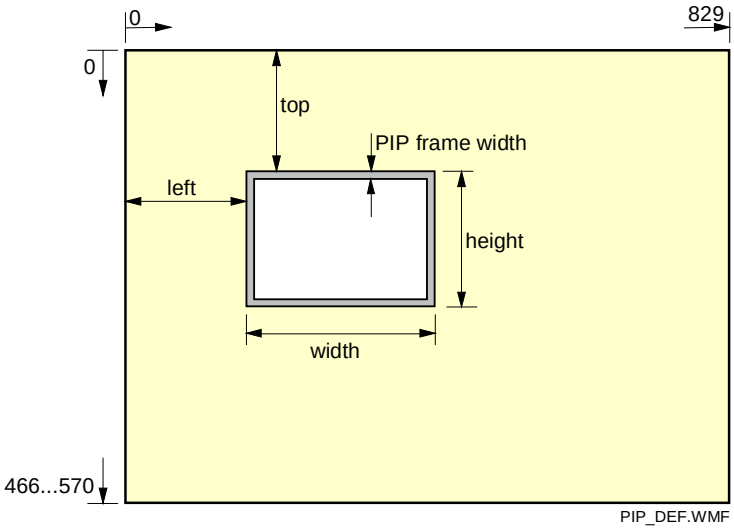


Fig. 96 Definition of the PIP-window

eight pixels. The PIP-window is defined at the input of the SAA4979, i. e. in the 27 MHz clock domain, however the frame around it is generated in the back-end of the IC in the 32 MHz domain. Therefore the PIP-window position values are not equal to the PIP-frame position values. They differ by a constant offset in horizontal and vertical direction and a constant multiplier in horizontal direction of 848 pixels to 720 pixels (according to the sample rate conversion from 27 MHz to 32 MHz) for each pixel. The PIP-interface manages all the necessary PIP-frame calculations to make the frame fit to the PIP position. Due to rounding errors in PIP-frame calculation and the internal storage as an integer data type it is possible to have a misplaced PIP-frame. This possible error is about ± one pixel in horizontal position and/or size. To fix this error, these values can be fine tuned by the register 'PIP_FRAME_TUNE'. Additionally the contents of the registers 'Horizontal- and Vertical-Delay' are automatically included in the calculation of the PIP-frame. If the PIP-window height is equal to the maximum size in the current display state and the PIP-window width exceeds 30% of the whole possible horizontal display size, the PIP-frame is limited to a 'Double Window Frame'. In this state only a left and right frame is visible. The width of this frame has an additional offset of seven ('horizontal_frame_width') to overwrite the ITU-data values from the main- and the sub-channel. If the 'Double Window Frame' shall look like a normal PIP-Frame, the bit 'PIP_DoubleWinBorder' must be set.

The master software communicates with the PIP-interface of the SAA4979 via I^2C-bus. Because the timing of the PIP-window generation in the SAA4979 must be synchronized to the PIP channel switching initiated by the master software, there is a handshake algorithm implemented in the SAA4979's firmware. In this way the master is able to check the right PIP-window generation for a correct PIP-channel switching. The following registers are provided in the firmware to control the PIP-interface.

9.2 PIP register definitions

9.2.1 Write Registers:

Subaddress $38: **PIP_Vtop** (bit 0..7 of 10)

Subaddress §39: **PIP_Hleft** (bit 0..7 of 10)

Subaddress $3A: **PIP_H_V_MSB**

7	6	5	4	3	2	1	0
PIP_Vhigh[8..9]		PIP_Hwidth[8..9]		PIP_Hleft[8..9]		PIP_Vtop[8..9]	

Subaddress $3B: **PIP_Hwidth** (bit 0..7 of 10)

Subaddress $3C: **PIP_Vheight** (bit 0..7 of 10)

Subaddress $3E: **PIP_Control**

7	6	5	4	3	2	1	0
PIP_ UV_Shift	PIP_ Double- WinBorder	PIP_ 60Hz	PIP_ Still	PIP_ NO_FC	PIP_ NO_TRC	PIP Win_sta	PIP Win_clr

PIP_UV_Shift

This bit allows to shift the PIP-window one pixel in order to compensate false colors due to UV misplacement.

0 . . PIP starts on even pixel number

1 . . PIP starts on odd pixel number

PIP_DoubleWinBorder

0 . . If the SAA4979's control software detects a 'Double Window' setting in the transmitted PIP values, a special PIP-frame is built. This frame has no top and bottom frame bar and the size of the left and right frame bar is increased to overwrite possible TRC-bytes in the ITU data stream of the main- and subchannel.

1 . . If the 'Double Window Frame' shall have the same appearance like a normal PIP-frame, this bit must be set.

PIP_60HZ

The control software of the SAA4979 knows the state of the incoming field frequency of the ITU main channel. In PIP mode the field frequency of the ITU sub channel can differ from the main channel. In all cases the control software must set the bits 'PIP_2FM_DC' and 'PIP_RASTER_CORR' to the right values to get full performance of the integrated raster correction. The master software must set this bit matching to the field frequency of the ITU sub channel before a PIP window is generated by the PIP interface of the BESIC422.

0 . . subchannel is 50 Hz

1 . . subchannel is 60 Hz

PIP_NO_FC

0 . . If this bit is not set, the SAA4979's control software generates the PIP-frame. The width can be set by the master software and is limited by the control software of the SAA4979 for a minimum value of two pixels. The register for the PIP-frame width has the sub-address $3D_{hex}$. If the ITU-input signal includes the TRC-code (timing reference code), the PIP-frame position and size values are set to disable the 'TRC-pixels' in the output. The TRC-code consists of four bytes. This is equal to two pixels in YUV 4:2:2 format. Because the PIP-frame generation takes place at the backend of the IC in the 32 MHz domain, there is an inaccuracy of the PIP-frame position and size calculation of at most one pixel (caused by the 'fixed sample rate conversation of 720 to 848 pixel). So the PIP-frame width should be set to a minimum value of three pixels. The color of the PIP-frame is the same as the color of the sidepanels and can be set by the registers 'sidepanel_color_uv' and 'sidepanel_color_y' (sub-address $2B_{hex}$ and $2C_{hex}$).

1 . . If this bit is set, the PIP-Interface does not automatically generate the PIP-frame. In this case the master software must set the right frame parameters.
If 'Vertical' and/or 'Horizontal' zoom or compress modes are active, this bit should be set. In this case the master software must generate the PIP-frame.

PIP_NO_TRC

0 . . TRC codes in the ITU-data stream are present.

1. . If this bit is set, the ITU-data stream into the SAA4979 does not include the ‘Timing Reference Code’ at the beginning and the end of each data line. So the PIP-window size must be reduced by the number of pixels that are equal to the size of the TRC-bytes. The TRC-code consists of four bytes. This is equal to two pixels in the YUV 4:2:2 format. With these values, the left position of the PIP-window is shifted by two pixels, and the ‘left + width’ position is shifted by two pixels. The control software of the SAA4979 does the right calculation of the PIP-window size.
Attention: if there are no TRC-bytes into the ITU-data stream, an error in the chrominance-displayed data can occur.

PIPWin_sta

If this bit is set, the PIP-interface software of the SAA4979 generates a new PIP-window. If the PIP-window position and size does not fit to the display state, an error code is set. Otherwise the PIP-ready bit is set. To generate a next PIP-window, the master software must reset this bit to zero and then set it again to one. If more then one PIP-window is to be displayed, the PIP-interface software automatically toggles between the PIP- and Multi-PIP functionality of the SAA4979. In this case the old display is frozen.
This bit is one of the two existing ‘handshake’ bits to synchronize the master software and the control software of the SAA4979 and can only be set or reset by the master software.

PIPWin_clr

If this bit is set, all active PIP-windows and all active PIP-frames are cleared. Only the active channel is visible on the whole screen. If this bit is set to zero and the ‘PIPWin_sta’ bit is also zero, the master software is able to control all PIP-functions of the SAA4979 without the use of the PIP-interface.

Subaddress $40: **PIP_Frame_Tune**

7	6	5	4	3	2	1	0
PIP_Fhigh_Tune		PIP_Fwidth_Tune		PIP_Fleft_Tune		PIP_Ftop_Tune	

With these bits the PIP-frame can be shifted horizontally and vertically, or the width and height can be changed. The following values are possible:

0 = no shift/change

1 = add shift/change one pixel

2 = add shift/change two pixel

3 = sub shift/change one pixel

The PIP-Window position stays fixed, only the frame position and/or size are changed.

9.2.2 Read Registers

Subaddress $01: **PIP_Status**

7	6	5	4	3	2	1	0
(res.)	(res.)	PIP height error	PIP width error	PIP left error	PIP top error	PIP_ Ready	x

x . . bit not PIP related

PIP_Ready

If this bit is set, the PIP-interface has built the PIP-window. Now the master software is able to reset the 'Start PIP-window' bit to zero and to switch the channel (handshake algorithm). Every time the master software reads this register, this bit is automatically reset to zero. This bit also belongs to the 'Error' bits. It is the second handshake bit to control the synchronization between the master software and the control software of the SAA4979 and can only be set or reset by the control software (I^2C-slave).

9.3 Additional information

The settings of 'Vertical Zoom' and / or 'Vertical Compression' have an influence on the PIP-Window size and position, but no influence on the PIP-frame size and position. The values of these registers should never change if a PIP-window is active. If these register values are not equal to zero before the PIP-window generation starts, the automatic PIP-frame generation must be disabled (see 'PIP_control' register and 'PIP_NO_FC' bit).

9.4 PIP-Window Example

All PIP-window generation must interact with the video-decoder in the subchannel (SAA7114 or SAA7118). This decoder scales the incoming video signal to the size of the PIP-window.

Example for nine PIP-Windows:

PIP number	PIP_Top (V_Start)	PIP_Left (H_Start)	PIP_MSB	PIP_Width	PIP_Height
1	10	20	00	BC	AC
2	C9	20	00	BC	AC
3	82	20	01	BC	AC
4	10	08	04	BC	AC
5	C9	08	04	BC	AC
6	82	08	05	BC	AC
7	10	F0	04	BC	AC
8	C9	F0	04	BC	AC
9	82	F0	05	BC	AC

(All values are hexadecimal)

10. I²C register tables (software version 4.4)

This version of the SAA4979 firmware is intended to control an MK14 or MK14-EM board equipped with SAA4993 (FALCONIC PLUS), SAA4994 (RAVEN) or SAA4998 (FALCONIC EM).

Firmware version: V 4.4
Date: March 21, 2003

The following tables describe the registers of the MK14-EM on-board microcontroller (here referred to as slave µC) which can be accessed via I²C bus by the main controller (I²C bus master controller).

When **writing** to the module the slave address is $\mathbf{68}_H$. A control sequence consists of at least three bytes:

[slave address] [write-subaddress] [data byte]

If writing to consecutive subaddresses is intended, only the first subaddress needs to be sent. Starting with the second data byte the associated subaddress in the slave µC is incremented automatically (auto-increment mode):

[slave address] [write-subaddress] [data byte 1] [data byte 2] . . . [data byte n]

When **reading** from the module the slave address is $\mathbf{69}_H$. A control sequence consists of two bytes:

[slave address] [read-subaddress]

After the read-subaddress is sent the IPQ module starts transmitting the data byte of the designated subaddress. After the master µC has sent an acknowledge signal the data byte of the next subaddress is transmitted (auto-increment mode). The transmission can be aborted by the master by omitting the acknowledge signal.

The following tables list the subaddress, the variable name, the bit position and a description of the function. The default value for the subaddress is given in parentheses in the description column. This is the value that was loaded to the register at startup.

10.1 Write registers

Sub-Addr. (hex)	Bit Pos	Variable Name	Description
FIELD_CONTROL			
$00			**Field_Control_1 ($70)**
	0	PSC	**Progressive scan mode** 0 = Progressive scan mode off 1 = Progressive scan mode
	1	PSC_DR	**Mode for PROGRESSIVE SCAN** 0 = normal mode 1 = in PROGRESSIVE SCAN mode the display will run in interlace mode
	2	PSC_1080i_60p	**Progressive Scan 1080-i Mode (PSC, AFF need to be set)** 0=off 1=on (60Hz input signal)
	3	PSC_1080i_50to60p	**Progressive Scan 1080-i Test Mode** **(only in combination with sub 0: PSC=1 and sub 1A, bit5: PAL_60p=1)** 0 = off 1 = on (50Hz input signal)
	4	A_MOVIE	**Automatic movie source detection** 0 = movie detection disable 1 = automatic movie source detection activated; in case a movie mode is detected, a movie will be processed (MOVIE, MOVIE_PHASE are readable via STATUS register)
	5	IM	**Incredible Motion Mode** 0 = Incredible Motion mode off 1 = Incredible Motion mode on
	6	LFR	**Line flicker reduction** 0 = Line Flicker Reduction mode off 1 = line Flicker Reduction mode on
	7	MOVIE_FALLBACK	**Fallback mode** in Movie 2_2 and Movie 3_2 processing, Fallback-threshold (GlobalACTmsb) programmable via subaddress $1B and $1C. 0 = Fallback disabled (default) 1 = Fallback enabled (see also threshold at subad. $1B and $1C)
$01			**Field_Control_2 ($00)**
	0	MOVIE	**Forced Movie mode** 0 = Forced Movie mode off 1 = Forced Movie mode on (ABAB, without NM)
	1	PHASE	**Forced phase flag to be set in combination with MOVIE** 0 = normal (ABAB) 1 = 180° phase shift (BCBC)
	2	STP	**Still picture mode** 0 = off (default) 1 = on (one field out of AABB, full frame median filtered out of LFR)

	3..5	HZOOM_COMP_PAN1	**Hzoom/Hcompression/Panorama** 0 = HZOOM_COMP_PAN2 (sub 02) active 1 = 14:9 Horiz. Compress (0.87) 2 = 16:9 Horiz. Compress (0.75) 3 = Panorama 4 = Amaronap 5 = 14:9 Horiz. Zoom (1.17) 6 = 16:9 Horiz. Zoom (1.33) 7 = 22:9 Horiz. Zoom (1.83)
	6..7	PP_COMP	**Picture Position HCompress** 0 = centre 1 = max left 2 = max right
$02			**Peaking_direct1 ($00)**
	0..2	alpha	**Peaking alpha (enable via sub 28, bit 1)** 0 = 0 1 = 1/16 2 = 2/16 3 = 3/16 4 = 4/16 5 = 5/16 6 = 6/16 7 = 8/16
	3..5	beta	**Peaking beta (enable via sub 28, bit 1)** 0 = 0 1 = 1/16 2 = 2/16 3 = 3/16 4 = 4/16 5 = 5/16 6 = 6/16 7 = 8/16
	6..7	HZOOM_COMP_PAN2	**Hzoom/Hcompression/Panorama adjust** 0 = HZOOM3, sub 17hex (if HZOOM_COMP_PAN1 = 0, sub 01) 1 = Horiz. Compr. factor 0.94 (if HZOOM_COMP_PAN1 = 0) 2 = Horiz. Zoom factor 1.12 (if HZOOM_COMP_PAN1 = 0) 3 = Horiz. Zoom factor 1.25 (if HZOOM_COMP_PAN1 = 0)

DEINTERLACE SHIFT

$03			**Deinterlace_Shift ($00)**
	0	No_Deint_shift_Movie	**Enable Interlace Shift in Progressive Movie Mode** 0 = on 1 = off
	1..2	Deint_shift	**value of Deinterlace Shift Factor** 0 = 00hex 1 = 20hex 2 = 40hex 3 = 80hex

	3	G_Mode	**Generator mode** 0 = off 1 = on
	4	FSFM	**Forced Single Field** 0 = off 1 = on
	5	AFF	**Acquisition field frequency (50/60 Hz)** 0 = 50 Hz 1 = 60 Hz
	6	ScfViaHbln	**Screenfade Selection** 0 = Screenfade via sidepanels (default) 1 = Screenfade via H-Blanking Enable screenfade via subad. 05
	7	ScfLowSpeed	**Screenfade Speed** 0 = high speed (default) 1 = low speed

V_ZOOM

$04			**Vertical_Zoom ($00)**
	0..5	VZOOM	**Conversion factor** from 1 to 1.5 in steps of 1/32
	6	VCOMPR	**Vertical compress bit** (0 = default) 0 = vertical zoom (factor set by VZOOM) 1 = vertical compression (factor set by VZOOM)
	7	DIS_Feat	**Feature Mode** 0 = normal mode 1 = disable Feature Mode

NOISE REDUCTION & SCREEN FADE

$05			**NR_Screenfade ($80)**
	0..1	NR	**Noise reduction** 0 = off 1 = low 2 = middle 3 = high
	2	Auto_Noise	**Auto_Noise** 0 = off 1 = on
	3	UV_Slave	**Noise reduction of U and V slaved to Y** (FALCONIC/RAVEN) 0 = U and V not slaved to Y 1 = U and V slaved to Y
	4..5	SCF	**Screen fade** 0 = off 2 = fade in 3 = fade out

	6	DebugMode	**Debug Mode:** sidepanel color indicatedes mode: video = red movie23 = green movie22 = blue phase 22 = magenta. **Attention:** set bit SET_SIDEP ($0A) to enable the sidepanel and set the size of the sidepanels ($2D, $2E, $35) 0 = Debug Mode is disabled (default) 1 = use of sidepanels to show currently processed mode
	7	DME	**Detected Mode Evaluation:** compare the phase of current and previously detected movie mode 0 = mode evaluation is disabled 1 = mode evaluation is enabled (default) thresholds are programable via register $41, $42 and $43

WRITE ENABLE DELAY

$06			**HWE_Delay ($00)**
	0..7	HWE1F	HWE1 fine delay (offset) to default two pixel delay to blank the TRC-bytes of the incoming ITU data stream
$07			**VWE_Delay ($00)**
	0..6	VWE1D	VWE1 delay (Bit 0..6)
	7	A_VSHIFT	**VSHIFT for VZOOM** 0 = via VWE 1 = in FSFM auto

BLANK FIELDS

$08			**EDDI etc. ($00)**
	0..1	EDDICmp	factor to specify the size of the additional compensation area left and right of the real edge. A higher factor (eg. 1) can increase the compensation in regions far away from the true edge 0 = factor 1 1 = factor 1/2 2 = factor 1/4 3 = factor 1/8
	2	P15	**Port bit P1.5** 0=clear 1=set
	3	PIPDataDelay	**PIP Data Delay** 0 = input data delay will be delayed by one clock cycle with respect to WE2 (write enable); default ! 1 = no delay
	4	Auto_Movie_ Processing_off	**Disable Auto Movie Processing** 0 = default 1 = disable Auto Movie Processing Auto Movie Detection remains active (to be used in combination with moviephase_switched sub read 01)
	5	EDDI_SplitScreenDemo	**EDDI Split Screen Demo** 0 = Split Screen off 1 = Split Screen on

	6	Set_Bln	enable: hbln_sta ($30), hbln_sto ($31), vbln_sta ($32), vbln_sto ($33), hbln_vbln_MSB ($34)
	7	FormatFreeze	**Freeze Format** (Auto Format Detection) intended for PIP control 0 = default 1 = Freeze Format in Auto Format Detection

PORT SETTINGS

$09			**Port_Settings ($20)**
	0	Auto_Format _Detection	**Auto Format Detection** 0 = off 1 = on Enables Automatic Black bar detection and processing
	1	P12	**port bit P1.2 (only available without external memory access)** 0 = clear 1 = set
	2	P13	**port bit P1.3 (only available without external memory access)** 0 = clear 1 = set
	3	P14	**port bit P1.4** 0 = clear 1 = set
	4	TubeFormat_4_3	**Tubeformat Selection** (interacts with Auto_Format_Detection) 0 = 16:9 tube 1 = 4:3 tube
	5	NoiseEstRegOff	**Noise Estimation Register Control (REGs: 0x1E ... 0x27)** 0 = Enable direct Noise Est. Regs 1 = default
	6	UV_bandwidth _detection	**enable UV-bandwidth detection** 0 = default 1 = UV bandwidth detection on
	7	PictureShiftRight	**Picture Shift to the right** 0 = The picture is in normal H-position. 1 = if HZOOM_COMP_PAN1 (sub 01, 3..5) = 0 and HZOOM_COMP_PAN2 (sub 02, 6..7) = 0 and HZOOM3 (sub 17, 5..6) = 0 then the picture is shifted to the right side.

ENABLE BITS

$0A			**Enable_Bits ($00)**
	0	SET_HD_Shift	**Enable HD_shift** 0 = default HD settings are used 1 = HD shifted with offset via register $44
	1	SET_PLL	**Enable direct settings of PLL** ($36, $37)) 0 = disable direct settings (default) 1 = enable direct settings

	2	SAA4979_output_range	**Enable 10 bit range** for nominal output signal (in combination with bypass FSRC) 0 = 9 bit for nominal output signal; black level: 288 and white level: 727 1 = 10 bit for nominal output signal; black level: 64 and white level: 940
	3	bypass_FSRC	**Bypass Fixed Sample Rate Converter** (in combination with 10 Bit nominal output signal) 0= default, use FSRC 1= bypass FSRC
	4	SET_SIDEP	**Set sidepanel** position via I2C-bus (subaddress $2D, $2E, $35) 0 = normal mode 1 = start- stop via I²C-bus
	5	VEC_OVL	**Vector overlay** mode or colour output 0 = normal mode (colour output) 1 = vector overlay is enabled
	6	SET_NR	Enable bits for direct access of Noise Reduction control registers in BESIC422 or FALCONIC (see SEL_NR). 0 = Standard setting as chosen by NR0 and NR1 is active 1 = Enable direct access of Noise Reduction Registers
	7	SEL_NR	Selects IC for direct control of NR via direct access of control registers (when SET_NR = 1): 0 = BESIC422 (use I²C subaddresses 11h..17h) 1 = FALCONIC (use I²C subaddresses 11h ...14h and 18h...1Ah)

SETFIELD_VPEAK

$0B			**Setfield_Vpeak ($02)**
	0	PIP_Mode_SAA4993	**PIP Mode SAA4993** 0 = default (PIP via FEM memory, Raven DPCM mode during PIP) 1 = PIP Mode SAA4993, external PIP memories
	1	Fal_DoubleOutput_Off	**FalconicDoubleOutput** 0 = double output mode 1 = normal single output mode
	2	Clear_SetSafeShiFac	**Clear SetSafeShiFac** 0 = use SetSafeShiFac() 1 = do not use SetSafeShiFac()
	3	SetDirectRegsBBD	**SetDirectRegsBBD** 0 = default 1 = enables the Black Bar Detection Registers *event_value* and *slice level* (sub 45, 46), if AutoFormatDetection = 0
	4..7	V_PEAKING	**Vertical peaking** 4 = +6dB 3 = +5dB 2 = +3.5dB 1 = +2dB 0 = 0dB 15 = -2.5dB 14 = -6dB 13 = -12dB

AUXILIARY_DISPL_SIGNAL

$0C			**HADSta ($00)**
	0..7	HADSSTA_0_7	Start of hor. Auxiliary Displ. Signal LSB
$0D			**HADSto ($00)**
	0..7	HADSSTO_0_7	Stop of hor. Auxiliary Displ. Signal LSB
$0E			**HAD_VAD_MSB ($00)**
	0..1	HADSSTA_8_9	Start of hor. Auxiliary Displ. Signal MSB
	2..3	HADSSTO_8_9	Stop of hor. Auxiliary Displ. Signal MSB
	4..5	VADSSTA_8_9	Start of vert. Auxiliary Displ. Signal MSB
	6..7	VADSSTO_8_9	Stop of vert. Auxiliary Displ. Signal MSB
$0F			**VADSta ($00)**
	0..7	VADSSTA_0_7	Start of vert. Auxiliary Displ. Signal LSB
$10			**VADSto ($00)**
	0..7	VADSSTO_0_7	Stop of vert. Auxiliary Displ. Signal LSB

NOISE REDUCTION

$11			**KSTEP01 ($00)**
	0..7	KSTEP01	step in adaptive curve; weight 1 (enable via SET_NR, SEL_NR, sub 0Ah)
$12			**KSTEP23 ($00)**
	0..7	KSTEP23	step in adaptive curve; weight 2 (enable via SET_NR, SEL_NR, sub 0Ah)
$13			**KSTEP45 ($00)**
	0..7	KSTEP45	step in adaptive curve; weight 4 (enable via SET_NR, SEL_NR, sub 0Ah)
$14			**KSTEP67 ($00)**
	0..7	KSTEP67	step in adaptive curve; weight 8 (enable via SET_NR, SEL_NR, sub 0Ah)

NR_REG_BESIC422

$15			**KLUM_CTRL ($00)**
	0..3	klumafix	**klumafix** (k-factor of noise reduction for Y) (enable via SET_NR, SEL_NR, sub 0Ah)
	4..6	yadapt_gain	**yadapt gain** (enable via SET_NR, SEL_NR, sub 0Ah)
	7	lumafix	**lumafix** 0 = adaptive 1 = fixed (k-factor defined by klumafix) (enable via SET_NR, SEL_NR, sub 0Ah)
$16			**KCHROMA ($00)**
	0..3	kchromafix	**kchromafix** (k-factor of noise reduction for U and V) (enable via SET_NR, SEL_NR, sub 0Ah)
	4..6	cadapt_gain	**cadapt gain** (enable via SET_NR, SEL_NR, sub 0Ah)
	7	chromafix	**chromafix** 0 = adaptive 1 = fixed (k-factor defined by kchromafix) (enable via SET_NR, SEL_NR, sub 0Ah)

$17			**MISCELLANEOUS ($1F)**
	0	klumatochroma	**klumatochroma** 0 = off 1 = on (enable via SET_NR, SEL_NR, sub 0Ah)
	1	unfiltered	**unfiltered** 0 = off 1 = on (enable via SET_NR, SEL_NR, sub 0Ah)
	2	noiseshape	**noiseshape** 0 = off 1 = on (enable via SET_NR, SEL_NR, sub 0Ah)
	3	splitscreen	**splitscreen** 0 = off 1 = on (enable via SET_NR, SEL_NR, sub 0Ah)
	4	NREN	**NREN** 0 = off 1 = on (enable via SET_NR, SEL_NR, sub 0Ah)
	5..6	HZOOM3	**HZOOM_3** 0 = off (if HZOOM_COMP_PAN1 and ..2 = 0, sub 01, 02) 1 = 15:9 Panorama (if HZOOM_COMP_PAN1 and ..2 = 0) 2 = Hzoom 1,1 DW for FalconicEM (if HZOOM_COMP_PAN1 and ..2 = 0) 3 = Hzoom factor 2 (necessary for 3D mode)
	7	Mode3D	**Mode3D:** Main and Subchannel display is fieldwise toggled, 2 different camera positions may be superpositioned to display 3D frames 0 = default 1 = Mode 3D active (in combination with HZOOM3 = 3)

NR_REG_FALCONIC

$18			**Gain_fix_y ($00)**
	0..3	GainFix_Y	**Fixed gain for Y** (enable via SET_NR, SEL_NR, sub 0Ah)
	4..6	GainDif_Y	**Set gain in difference signal** for adaptive DNR Y 5 = 4 4 = 2 3 = 1 2 = 1/2 1 = 1/4 0 = 1/8 (enable via SET_NR, SEL_NR, sub 0Ah)
	7	FIXY	**Set gain behaviour for Y** 0 = adaptive 1 = fixed (enable via SET_NR, SEL_NR, sub 0Ah)

$19			**Gain_fix_UV ($00)**
	0..3	GainFix_UV	**Fixed gain for UV** (enable via SET_NR, SEL_NR, sub 0Ah)
	4..6	GainDif_UV	**Set gain in difference signal** for adaptive DNR UV 5 = 4 4 = 2 3 = 1 2 = 1/2 1 = 1/4 0 = 1/8 (enable via SET_NR, SEL_NR, sub 0Ah)
	7	FIXUV	**Set gain behaviour for UV** 0 = adaptive 1 = fixed (enable via SET_NR, SEL_NR, sub 0Ah)
$1A			**Dnr_misc ($08)**
	0..2	VecComp	**Set degree of hor. vector compensation in Y DNR** 0 = 0 1 = 1/8 2 = 2/8 3 = 3/8 4 = 4/8 5 = 5/8 6 = 6/8 7 = 7/8 (enable via SET_NR, SEL_NR, sub 0Ah)
	3	Noise_Shape	**Enable noise shaping** 0= Noise shaping off 1= Noise shaping on (default)
	4	PIP_25Hz_Strobo_Mode	**PIP 25Hz Strobo Mode** 0 = default 1 = activate PIP 25HZ Strobo Mode
	5	PAL_60p	**PAL 60 Hz progressive** 0 = default 1 = in combination with PSC (sub 00, bit 0) conversion of 50Hz PAL input signal into 60Hz progressive output signal
	6	DnrSplit	**Dnr split in reg. DnrColorMode (Falconic/Raven)** 0 = off 1 = on
	7	DnrHpOn	**DnrHpOn in reg. DnrColorMode (Falconic/Raven)** 0 = off 1 = on

MOVIE FALLBACK THRESHOLD

$1B			**MovieFallback_1_Threshold ($27)**
	0..7	MovieFallback_1 _Threshold	Movie_Fallback_Threshold for Mode 1 (with less motion compensation in movie23 or true movie mode in movie22); if this value is $00 then mode is always active; if value is $FF then mode is disabled; Mode must always be enabled by the bit 'MOVIE_FALLBACK' in subaddress $00!

$1C			**MovieFallback_2_Threshold ($3A)**
	0..7	MovieFallback_2 _Threshold	Movie_Fallback_Threshold for Mode 2 (even less motion compensation in movie22; without motion compensation, TRUE-movie mode, in movie23); if this value is $00 (and sub 1B = 00) then mode is always active; if value is $FF then mode is disabled; Mode must always be enabled by bit 'MOVIE_FALLBACK' in subaddress $00! **Attention!** always choose a higher value in sub 1C than in sub 1B

MUX12 FRONTEND

$1D			**BESIC422_MUX ($07)**
	0	Select_Data_Input1	**Select data source for main channel** 0 = select channel 2 1 = select channel 1
	1	uv_sign1	**Toggle UV sign channel 1** 0 = off 1 = on
	2	uv_sign2	**Toggle UV sign channel 2** 0 = off 1 = on
	3	ForcePLLopen _if_FeatureMode	**Force_PLLopen_if_FeatureMode** 0 = off 1 = If Feature Mode = 1 => PLL open = 1
	4	SAA7118_Foet_is_set	**SAA7118_FOET** 0 = Forced odd/even toggle bit is cleared 1 = Forced odd/even toggle bit is set
	5	PSC_Raven _majority_select_off	**PSC Raven Majority Selection Off** 0 = default (PSC Raven majority selection on) 1 = PSC Raven majority selection off
	6		reserved; - to be cleared
	7	Split_Screen_MC	**Split Screen Motion Compensation** 0 = off 1 = Vertical Split Screen: left side uncompensated, right side motion compensated

NOISE ESTIMATION WRITE

$1E			**compns_ypscale ($01)**
	0..1	ypscale	Y scaling factor for prefilter
	2..5	compensate	compensate value can be added to NEST
	6	SetSfr	set field recognition 0 = default 1 = toggle field recognition
	7	SetDrABAB	set display raster ABAB 0 = default 1 = set display raster ABAB for ABAB movie mode
$1F			**gain_sob_clip_ne ($01)**
	0..2	gain_upbnd	**Set gain** of upper boundary
	3	sob_negl	**set SOB neglect external value** 0 = off 1 = on

	4	sel_sob_negl	**SOB neglect value** 0 = internal value 1 = external value
	5..6	clip_offs	**Clip offset** for selection of blocks
	7	Mode_75i	**75/90 Hz** interlace display mode 0 = off (default) 1 = 75/90 Hz interlace display mode (forced)
$20			**wanted_value ($BE)**
	0..7	wanted_value	No. of estimates searched in the interval closest to 0
$21			**lb_detail ($32)**
	0..7	lb_detail	Lower boundary detail for the absolute difference ADif
$22			**upb_detail ($BE)**
	0..7	upd_detail	Upper boundary detail for the absolute difference ADif
$23			**hm_Win_Sta ($06)**
	0..7	window_hstart	horizontal measurement window start
$24			**hm_Win_Sto ($B4)**
	0..7	window_hstop	horizontal measurement window stop (720 pixel / 4 = 180 pixel)
$25			**vm_Win_Sta ($1E)**
	0..7	window_vstart_0_7	vertical measurement window start LSB
$26			**vm_Win_Sto ($0E)**
	0..7	window_vstop_0_7	vertical measurement window stop LSB (shoud be less then maximum numbers of lines, e.g. PAL = 312)
$27			**vm_Win_MSB ($02)**
	0	window_vstart_8	vertical measurement window start MSB
	1	window_vstop_8	vertical measurement window stop MSB
	2	EggSlcThr_0	set Falc/Raven EggSlcThr via sub 2F, bit 7
	3	EggSlcThr_1	set Falc/Raven EggSlcThr via sub 2F, bit 7
	4	EggSlcThr_2	set Falc/Raven EggSlcThr via sub 2F, bit 7
	5	EggSlcThr_3	set Falc/Raven EggSlcThr via sub 2F, bit 7
	6	EggSlcThr_4	set Falc/Raven EggSlcThr via sub 2F, bit 7
	7	EggSlcThr_5	set Falc/Raven EggSlcThr via sub 2F, bit 7

DYNAMIC H-PEAKING

$28			**Fixed_And_Auto_Peaking ($14)**
	0	Auto_Peaking	**Auto_Peaking** 0 = off 1 = Dynamic Hpeaking on (based on selected Hpeaking curve, bits 2…5)
	1	Direct_Peaking	**Direct_Peaking** 0 = default, fixed curve or Auto_Peaking 1 = Use direct Peaking settings given in subaddress 02, 28 (bits 6,7) and 37
	2..5	HPeaking_Curve	**HPeaking_Curve** 0 = off 1…F= curves 1…15

	6..7	CoringThreshold	**CoringThreshold** 0 = low 1 = medlow 2 = medhigh 3 = high

DCTI

$29			**gain_threshold ($A4)**
	0..2	gain	**DCTI gain** 0 = 0 1 = 1 2 = 2 3 = 3 4 = 4 5 = 5 6 = 6 7 = 7
	3..6	threshold	**DCTI threshold**
	7	dcti_ddx_sel	**DCTI ddx_sel** 0 = low 1 = high
$2A			**DCTI_miscellan ($3A)**
	0..1	dcti_limit	**DCTI limit** 0 = 0 1 = 1 2 = 2 (default) 3 = 3
	2	dcti_sep	**DCTI sep** (separate processing of U and V) 0 = off (default) 1 = on
	3	dcti_protect	**DCTI protec** (hill protection) 0 = off 1 = on (default)
	4	dcti_postfilter	**DCTI postfilter** 0 = off 1 = on (default)
	5	dcti_superhill	**DCTI superhill** (super hill protection) 0 = off 1 = on (default)
	6		reserved
	7	Disable_DISPVPOS_ 120 Hz	**Disable DISPVPOS 120 Hz** 0 = default 1 = no adjustment of vert. position of display to acquisition at 120 Hz via VWE1D

POSTPROCESSING

$2B			**sidepanel_color_uv ($00)**
	0..3	sidep_color_u	sidepanels color overlay U 4 upper bits (2's complement)
	4..7	sidep_color_v	sidepanels color overlay V 4 upper bits (2's complement)

$2C			**sidepanel_color_y ($48)**
	0..7	sidep_y	sidepanels color overlay Y
$2D			**sidepanel_sta ($00)**
	0..7	sidepanel_start_2_9	sidepanel start position MSB (enable via SET_SIDEP, sub 0Ah)
$2E			**sidepanel_sto ($00)**
	0..7	sidepanel_stop_2_9	sidepanel stop position MSB (enable via SET_SIDEP, sub 0Ah)
$2F			**output_ctrl ($30)**
	0..3	y_delay_out	**y_delay_out**
	4	Post_uv_inv	**Post_uv_inv** 0 = default 1 = invert UV input
	5	y_dac_current_4uA	**y_dac_current_4uA** 0 = 2uA/bit 1 = 4uA/bit (Default)
	6	Toggle100HzDR_AABB	toggle 100Hz wise the display raster AABB 0 = default 1 = toggle display raster bit DR_AABB 100/120 Hz wise
	7	Enable_EggSlcThr_Ctrl	**EggSlcThr Control** 0 = off 1 = enable EggSlcThr control via sub 27, bits 2…7
$30			**hbln_sta ($41)**
	0..7	hbln_sta_0_7	horizontal blanking start LSB (829 pixel) (enable via Set_Bln, sub 08h)
$31			**hbln_sto ($05)**
	0..7	hbln_sto_0_7	horizontal blanking stop LSB (enable via Set_Bln, sub 08h)
$32			**vbln_sta ($33)**
	0..7	vbln_sta_0_7	vertical blanking start LSB (283/567 lines) (enable via Set_Bln, sub 08h)
$33			**vbln_sto ($17)**
	0..7	vbln_sto_0_7	vertical blanking stop LSB (enable via Set_Bln, sub 08h)
$34			**hbln_vbln_MSB ($13)**
	0..1	hbln_sta_8_9	horizontal blanking start MSB (enable via Set_Bln, sub 08h)
	2..3	hbln_sto_8_9	horizontal blanking stop MSB (enable via Set_Bln, sub 08h)
	4..5	vbln_sta_8_9	vertical blanking start MSB (enable via Set_Bln, sub 08h)
	6..7	vbln_sto_8_9	vertical blanking stop MSB (enable via Set_Bln, sub 08h)
$35			**y_NLP_SIDEP_LSB ($06)**
	0..1	NLP_lambda	NLP_lambda
	2..3	NLP_micro	NLP_micro
	4..5	sidepanel_start_0_1	sidepanel start position LSB (enable via SET_SIDEP, sub 0Ah)
	6..7	sidepanel_stop_0_1	sidepanel stop position LSB (enable via SET_SIDEP, sub 0Ah)

PLL

$36			**PLL_ck_cd ($07)**
	0..2	PLL_cd	**cd-value PLL** (damping factor) (enable via SET_PLL, sub 0Ah)
	3..7	PLL_ck	**ck-value PLL** (time constant) (enable via SET_PLL, sub 0Ah)
$37			**PLL_init ($00)**
	0..2	tau	**peaking tau** (enable via sub 28, bit 1) 0 = 0 1 = 1/16 2 = 2/16 3 = 3/16 4 = 4/16 5 = 5/16 6 = 6/16 7 = 8/16
	3..4	delta	**peaking delta** (enable via sub 28, bit 1) 0 = 0 1 = 1/16 2 = 2/16 3 = 4/16
	5..6	neggain	**peaking neggain** (enable via sub 28, bit 1) 0 = 0 1 = 1/16 2 = 2/16 3 = 4/16
	7	PLL_open	**PLL_open** 0 = PLL closed (default) 1 = PLL open (enable via SET_PLL, sub 0Ah)

PIP INTERFACE

$38			**PIP_Vtop ($00)**
	0..7	PIP_Vtop_0_7	Vertical position of the upper left corner of PIP LSB **If MainChannel_PIP (Bit 7 in Reg 3F) is set:** Bit7 = 0: MainPip position left Bit7 = 1: MainPip position right
$39			**PIP_Hleft ($00)**
	0..7	PIP_Hleft_0_7	Horizontal position of the upper left corner of PIP LSB **If MainChannel_PIP (Bit 7 in Reg 3F) is set:** Bit0…7 = PIP Hshift
$3A			**PIP_H_V_MSB ($00)**
	0..1	PIP_Vtop_8_9	Vertical postion of the upper left corner of PIP
	2..3	PIP_Hleft_8_9	Horizontal position of the upper left corner of PIP
	4..5	PIP_Hwidth_8_9	Width of PIP window
	6..7	PIP_Vhigh_8_9	Height of PIP window
$3B			**PIP_Hwidth ($00)**
	0..7	PIP_Hwidth_0_7	Width of PIP window LSB
$3C			**PIP_Vhigh ($00)**
	0..7	PIP_Vhigh_0_7	Height of PIP window LSB

$3D			**PIP_border_frame_width ($01)**
	0..3	vertical_frame_width	H frame width = 0 => H,V frame until picture edge
	4..7	horizontal_frame_width	V frame width = 0, H frame width > 0 => panel window
$3E			**PIP_Control ($00)**
	0	PIPWin_clr	clear all PIP windows
	1	PIPWin_sta	start PIP_Window
	2	PIP_NO_TRC	PIP_NO_TRC 0 = TRC bytes in ITU data stream (default) 1 = no TRC bytes in ITU datastream
	3	PIP_NO_FC	PIP_NO_FC 0 = automatic PIP-Frame control (default) 1 = PIP-Window without frame
	4	PIP_Still	Freeze contents of current PIP window (only possible, if 'Multi PIP' is disabled) 0 = PIP_Still is disabled (default) 1 = PIP_Still is enabled (no action if 'Multi PIP' is active)
	5	PIP_60HZ	field frequency from PIP-sub channel 0 = sub channel 50Hz (default) 1 = sub channel 60Hz
	6	PIP_DoubleWinBorder	change border in 'Double Window' mode 0 = only color bar in the middle of the window (default) 1 = whole frame equal to PIP mode
	7	PIP_UV_Shift	Shift PIP-window one pixel to compensate UV-misplacement 0 = first PIP pixel on even pixel number 1 = first PIP pixel on odd pixel number
$3F			**PIP_Special ($01)**
	0	PIP_AutoSet	Control of 'PIP_2_FIELD', 'PIP_FM_DC' and 'PIP_R_CORR' 0 = manual settings via I2C 1 = automatic settings (default)
	1	PIP_2FIELD	**PIP field mode** 0 = one field mode 1 = two field mode
	2	PIP2_FM_DC	**Field memory compensation** 0 = simple mode 1 = compensation on
	3	PIP_R_CORR	Phase relation of **PIP raster correction** 0 = off 1 = on
	4	PIPonlyFrameControl	**Automatic control of all PIP registers** over PIP interface or only PIP frame control 0 = automatic control over PIP interface (default) 1 = map REG38..3C to PIP frame **Attention!** If this value is not equal zero, all PIP interface functionality is disabled REG38: Vtop (original: Vtop) REG39: Hleft (original: Hleft) REG3A: H_V_MSB (original: H_V_MSB) REG3B: Hright (original: Hwidth) REG3C: Vbottom (original: Vhigh)

	5	PIPdynamicShift	**enable dynamic shift of PIP-window** 0 = dynamic shift is disabled (default) 1 = dynamic shift is enabled In this mode the position and size of the current PIP-window and frame can be dynamically changed (NOT for Multi-PIP!)
	6	OldPIPAutoSet	**control the 'PIP_AutoSet' settings** [PIP_2FIELD, PIP2_FM_DC, PIP_R_CORR] 0 = use new 'PIP_AutoSet' settings (default) 1 = use old 'PIP_AutoSet' settings
	7	MainChannel_PIP	**Main_Channel_PIP** 0 = off 1 = Main Channel PIP active
$40			**PIP_FRAME_TUNE ($00)**
	0..1	PIP_Ftop_Tune	change **position of PIP-frame top** 0 = no change of Frame-position 1 = +1 pixel 2 = +2 pixel 3 = -1 pixel
	2..3	PIP_Fleft_Tune	change **position of PIP-frame left** 0 = no change of Frame-position 1 = +1 pixel 2 = +2 pixel 3 = -1 pixel
	4..5	PIP_Fwidth_Tune	change **size of PIP-frame width** 0 = no change of Frame-position 1 = +1 pixel 2 = +2 pixel 3 = -1 pixel
	6..7	PIP_Fhigh_Tune	change **size of PIP-frame high** 0 = no change of Frame-position 1 = +1 pixel 2 = +2 pixel 3 = -1 pixel

MOVIE MODE EVALUATION

$41			**movieSwitchSettings_1 ($86)**
	0..3	MinMovieTime	Allowed minimum value of the movie processing time, will be compared during video mode with the last movie processing time measured by the µC. (resolution is 2/10s)
	4..7	MinVideoTimeNTSC	Allowed minimum value of the video processing time in NTSC, will be compared during video mode with the last video processing time measured by the µC. The comparison is done in NTSC mode whenever the last movie processing time is shorter than MinMovieTime. (resolution is 2/10s)
$42			**movieSwitchSettings_2 ($88)**
	0..3	VideoToMovieSwitch PenaltyPAL	Penalty time for an intended video to movie mode switch after detection of a short movie processing time (< MinMovieTime). (resolution is 2s)

	4..7	VideoToMovieSwitch PenaltyNTSC	Penalty time for an intended video to movie mode switch after detection of a short movie processing time (< MinMovieTime) and also a short last video processing scene (< MinLastVideoTimeNTSC). (resolution is 2/10s)
$43			**movieSwitchSettings_3 ($68)**
	0..3	wrongPhasePenalty	Penalty time for an intended video to movie mode switch after detection of a phase change. (resolution is 4s)
	4..7	EddiFil_Proscan8	minimal required edge filter value at start and end of the monotonous region to be a reliable edge point; will be automatically incremented by the µC in pictures with noise (add on values Est_Noise 0…5) (values 0…60 in multiples of 4, max input value allowed: 0x0A)

HD SHIFT

$44			**HD_offset ($FC)**
	0..7	HD_offset	Offset for HDSTA and HDSTO [4 pixel resolution] (must be enabled by bit 'SET_HD_shift' in register $0A)

BLACK BAR DETECTION

$45			**bbd_event_value_reg**
	0..5	bbd_event_val	**Blackbar detection event** value via SetDirectRegsBBD (sub 0B)
	6	Hold_New_HDsto	**Hold new Hdsto value** programmed via PIP_Vtop and New_Hdsto_from_PIPVtop=1 0 = Hdsto default or new via PIPVtop (dep. on New_Hdsto_from_PIPVtop 1 = Hold new Hdsto value (PIPVtop may be used for PIP)
	7	New_HDsto_from_ PIPVtop	**Get new HDsto value** from PIPVtop 0 = Hold Hdsto or Default Hdsto (dep. on Hold_New_Hdsto) 1 = Use Hdsto value as given in PIPVtop register (subaddress 38 hex)
$46			**bbd_slice_level_MSB**
	0..5	bbd_slicing_level	slice level (*2) via SetDirectRegsBBD (sub 0B)
	6..7	EddiLng_Proscan8	minimal required length of monotonous region to be reliable; higher values results in higher reliability of EDDI, but less steep edges will be detected 0 = 2 pixels 1 = 3 pixels 2 = 4 pixels 3 = 5 pixels
$47			**HWinSta ($54)**
	0..7	HWINSTA	Horizontal Window Start
$48			**HWinSto ($BE)**
	0..7	HWINSTO	Horizontal Window Stop
$49			**VWinSta ($15)**
	0..7	VWINSTA	Vertical Window Start
$4A			**VWinSto ($3C)**
	0..7	VWINSTO	Vertical Window Stop

$4B			**Proscan7 ($30)**
	0..1	EddiMR	factor for the comparison of the monotonous regions belonging to 2 edge points to verify an edge 0 = factor 1 1 = factor 1/2 2 = factor 1/4 3 = factor 1/8
	2..3	EddiED	factor for the comparison of the monotonous regions belonging to 2 edge points and the edge point distance to verify an edge 0 = factor 1 1 = factor 1/2 2 = factor 1/4 3 = factor 1/8
	4..7	EddiDif	minimal required Y difference at the edge point position to be a reliable edge point; higher values result in higher reliability of EDDI, but less edges will be detected (values 0…60 in multiples of 4)
$4C			**Proscan9 ($08)**
	0..3	EddiOfs	offset to increase or decrease the amount of EDDI compensation; lower values increase the amount of compensation (1 to 16)
	4..7	EddiLim	limitation of the compensation factor of EDDI; 1 limits to full EDDI compensation, 16 limits to almost no EDDI compensation (1 to 16)

WRITE FACTORY SETTINGS

$99			**WriteFactorySettings ($18)**
	0	SetNewSteepness Window	Set Steepness Window via $47,48,49,4A 0 = default steepness window (BBD dependant) 1 = fixed steepness window: hwinsta = $47 hwinsto = $48 vwinsta = $49 vwinsto = $4A
	1	Quit_Reset_Write	if set, then the master software has acknowledged the RESET condition. In this case the bit 'Show_Reset_Write' was reset to zero by the control software
	2	Enable_HW_Concept _Write	Enable direct settings of hardware concept via bits 3..4 of this register
	3..4	Set_Hardware_Concept _Write	Set hardware concept 0 = use automatic detected value (default) 1 = force BESIC422 only 2 = force BESIC422 / RAVEN 3 = force BESIC422 / FALCONIC
	5	EDDI_off	**EDDI (Edge Dependent Deinterlacer) control in FALCONIC EM** 0 = EDDI on 1 = EDDI off
	6	small_featmode_window	use small feature mode processing window 0 = allow +/- 4 halflines deviation (= V105) (default) 1 = allow +/- 2 halflines deviation in no. of lines per field
	7	Vres_Dis	Disable vertical reset in memory controller 0 = vertical reset enabled (default) 1 = vertical reset disabled

10.2 Read Registers

Sub-Addr. (hex)	Bit Pos	Variable Name	Description
STATUS			
$00			**STATUS ($40)**
	0	NON_IL	non interlace mode 0 = not active 1 = active
	1	FEATURE_MODE	feature mode 0 = not detected 1 = detected
	2	Im_Active	Incredible motion 0 = normal mode 1 = Incredible motion active
	3	MOVIE_FLAG	Movie 0 = no movie source detected 1 = movie source detected
	4	PHASE_FLAG	movie phase 0 = in phase 1 = not in phase
	5	SCREEN_FADE _ACTIVE	screenfade 0 = screen fade not active 1 = screenfade active
	6	READY	Ready to accept IIC commands 0 = not ready 1 = ready
	7	WATCH	Watchdog bit; will be toggeld when status byte is read by master uC, initialized with 0
MOVIE STATUS			
$01			**MOVIE_STATUS ($00)**
	0	MOVIE_MOD_FLAG	Movie mode flag 0 = 2:2 pulldown mode 1 = 2:3 pulldown mode
	1	PIP_READY	PIP is ready 0 = not ready 1 = ready
	2..5	PIP_error_code	PIP error code 0 = no error 1 = top line error 2 = left column error 4 = width error 8 = high error
	6	moviephase_switched	movie phase has switched (to be used if Auto_Movie_Processing_off = 1, subaddr. 08, bit 4) 0 = no switch (default) 1 = movie phase has switched
	7		reserved

SOFTWARE VERSION

$02			**SOFTWARE_VERSION ($44)**
	0..7	SOFTWARE_VERSION	version of the slave control software

HARDWARE ID

$03			**Detected_Hardware ($FB)**
	0..7	Detected_Hardware	Hardware ID. (50ms < concept detection < 650ms); the following concepts are supported: EBh = SAA4979 / SAA4994 (Raven) FBh = SAA4979 / SAA4993 (FalconicPlus) **Attention:** the master uC should wait 700 ms after power on before reading Detected_Hardware

NOISE ESTIMATION

$04			**noise_estimation ($00)**
	0..3	nest	noise estimation
	4..7		reserved
$05			**noise_estimation_filter ($00)**
	0..7	nest_filt	noise estimation filter
$06			**detail_counter_MSB ($00)**
	0..7	detail_cnt_h	detail counter MSBs
$07			**detail_counter_LSB ($00)**
	0..7	detail_cnt_l	detail counter LSBs
$08			**grey_counter ($00)**
	0..7	grey_cnt	grey counter

FORMAT

$09			**read_format ($00)**
	0..1	format	format (for Auto_Format_Detection = 1) 0 = no black bars 1 = 14:9 2 = 16:9 3 = 22:9
	2..7		reserved

SAA4993 READ REGISTERS

$0A			**Safe_FbLine ($00)**
	0..7	SafeFbLine	SNERT address 0C9, page 29
$0B			**HiAct_Cnt ($00)**
	0..7	HiActCnt	SNERT address 0D4, page 29

BLACK BAR DETECTION

$0C			**bbd_first_video_line ($00)**
	0..6	bbd_first_vid_line	blackbar detection number of first video line top/bottom search
	7	MSB_bbd _last_video_line	MSB 8(9) bbd_last_vid_line
$0D			**bbd_last_video_line ($00)**
	0..7	bbd_last_video_line	blackbar detection number of last video line top/bottom search, lower 8(9)

BAND WIDTH DETECTION

$0E			**UV_bandwidth_detect ($00)**
	0..7	UV_bandwidth_detect	UV bandwidth detection

DYNAMIC H-PEAKING

$0F			**steepness_max ($00)**
	0..7	steepness_max	steepness read register

READ FACTORY SETTINGS

$99			**ReadFactorySettings ($00)**
	0	Show_Reset_Read	If set a hardware reset was carried out. This bit was reset to zero if the bit Quit_Reset_Write was set by the master software (Handshake)
	1		reserved
	2	Enable_HW_Concept _Read	if set, then the hardware concept is set by the bits 3..4 of this register
	3..4	Set_Hardware_Concept _Read	Show the forced hardware concept 0 = use automatic detected value (default) 1 = force BESIC422 only 2 = force BESIC422 / RAVEN 3 = force BESIC422 / FALCONIC
	5..7		reserved

11. Appendix

11.1 Circuit diagrams of the IPQ module MK14-EM (H02VS08)

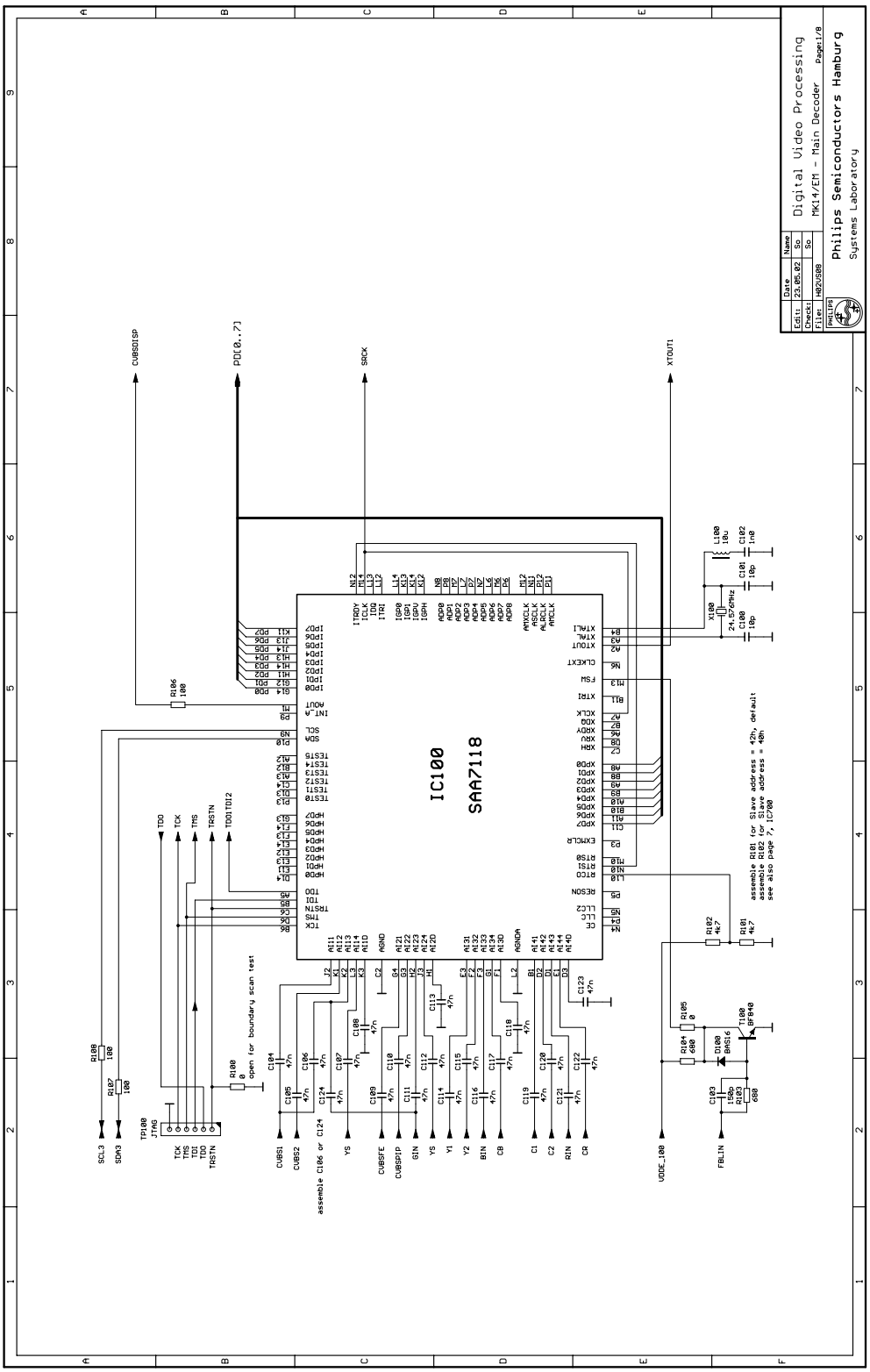


Fig. 97 IPQ module MK14-EM circuit diagram: sheet 1

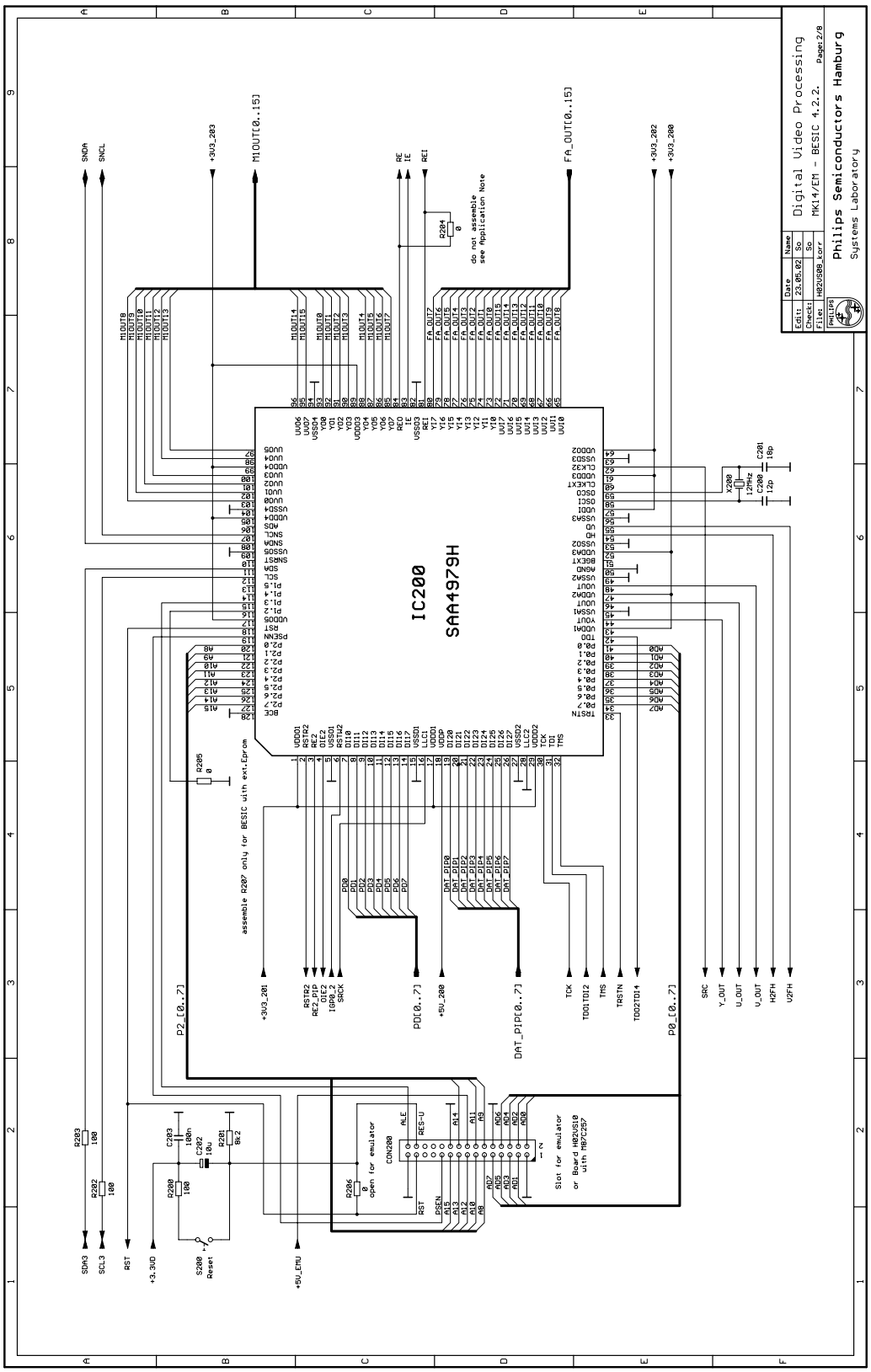


Fig. 98 IPQ module MK14-EM circuit diagram, sheet 2

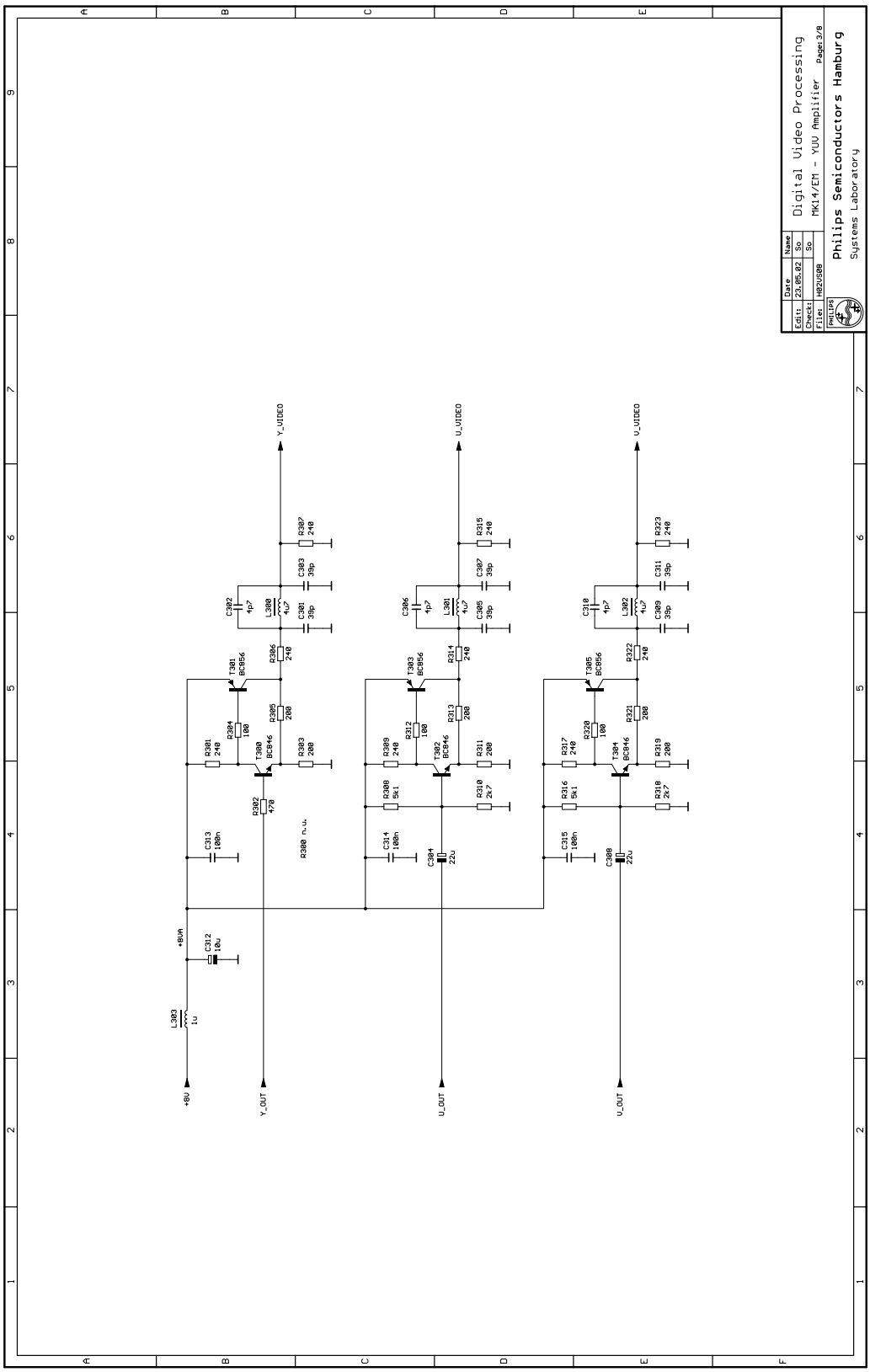


Fig. 99 IPQ module MK14-EM circuit diagram: sheet 3

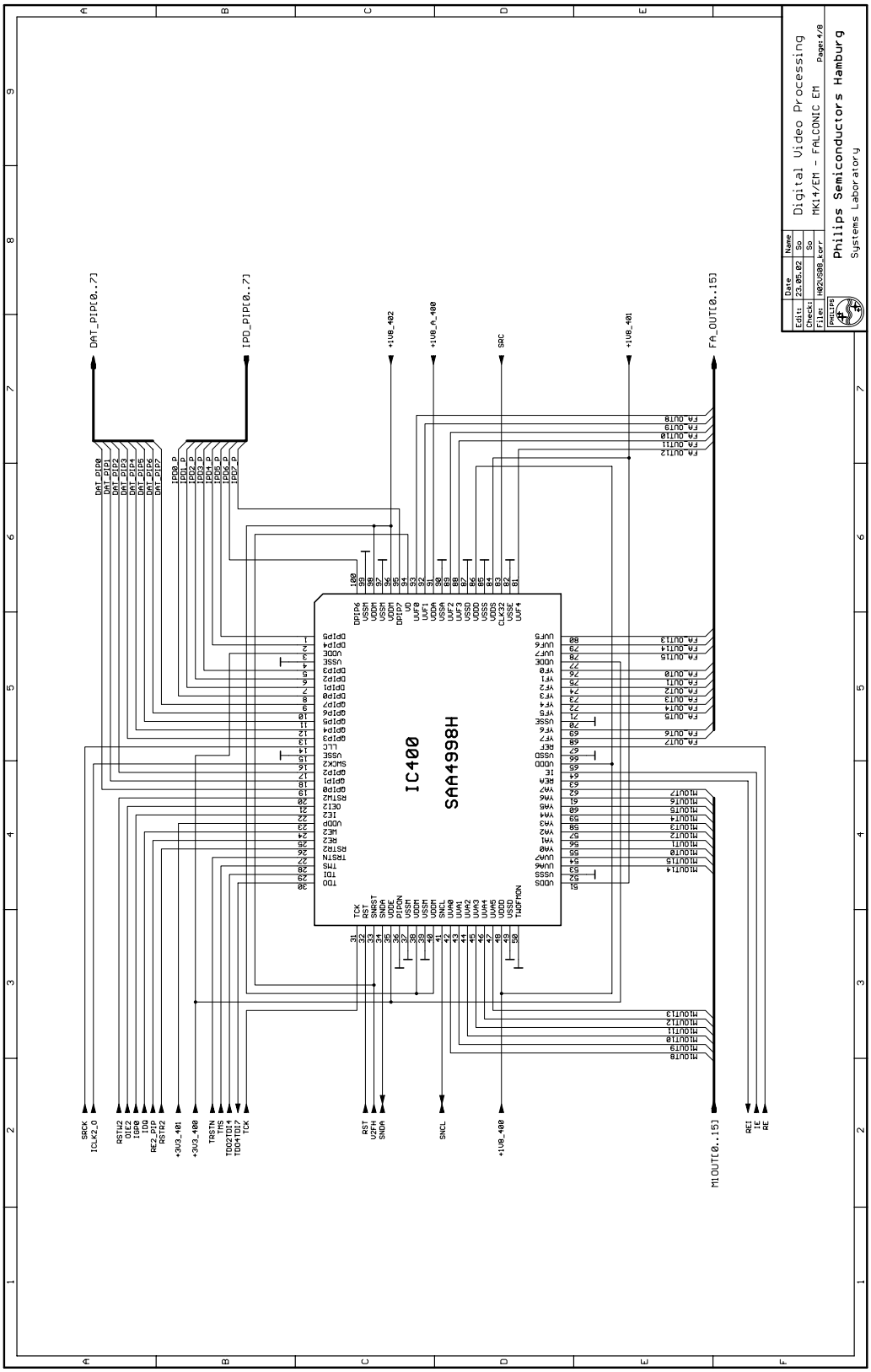


Fig. 100 IPQ module MK14-EM circuit diagram: sheet 4

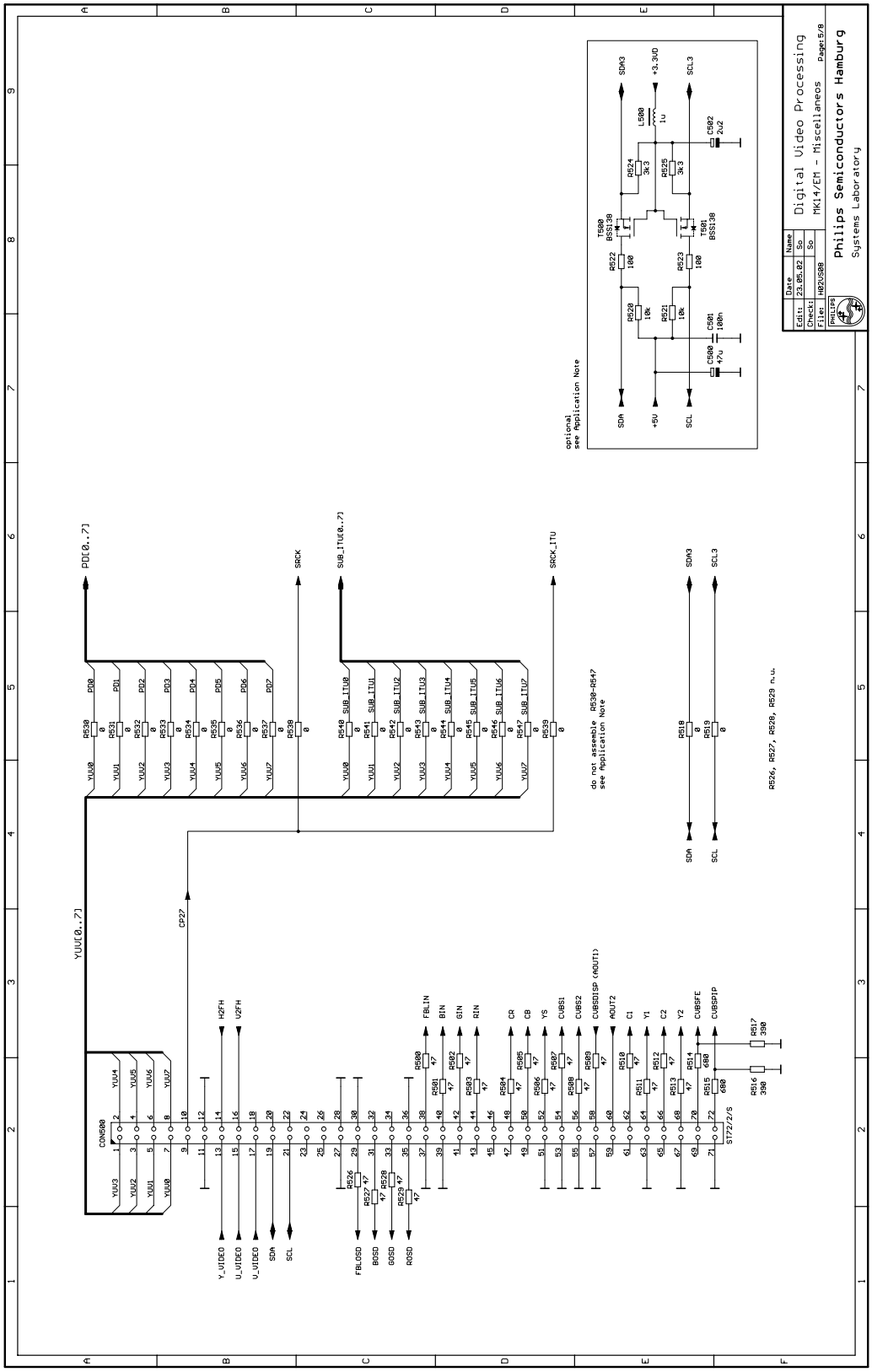


Fig. 101 IPQ module MK14-EM circuit diagram: sheet 5

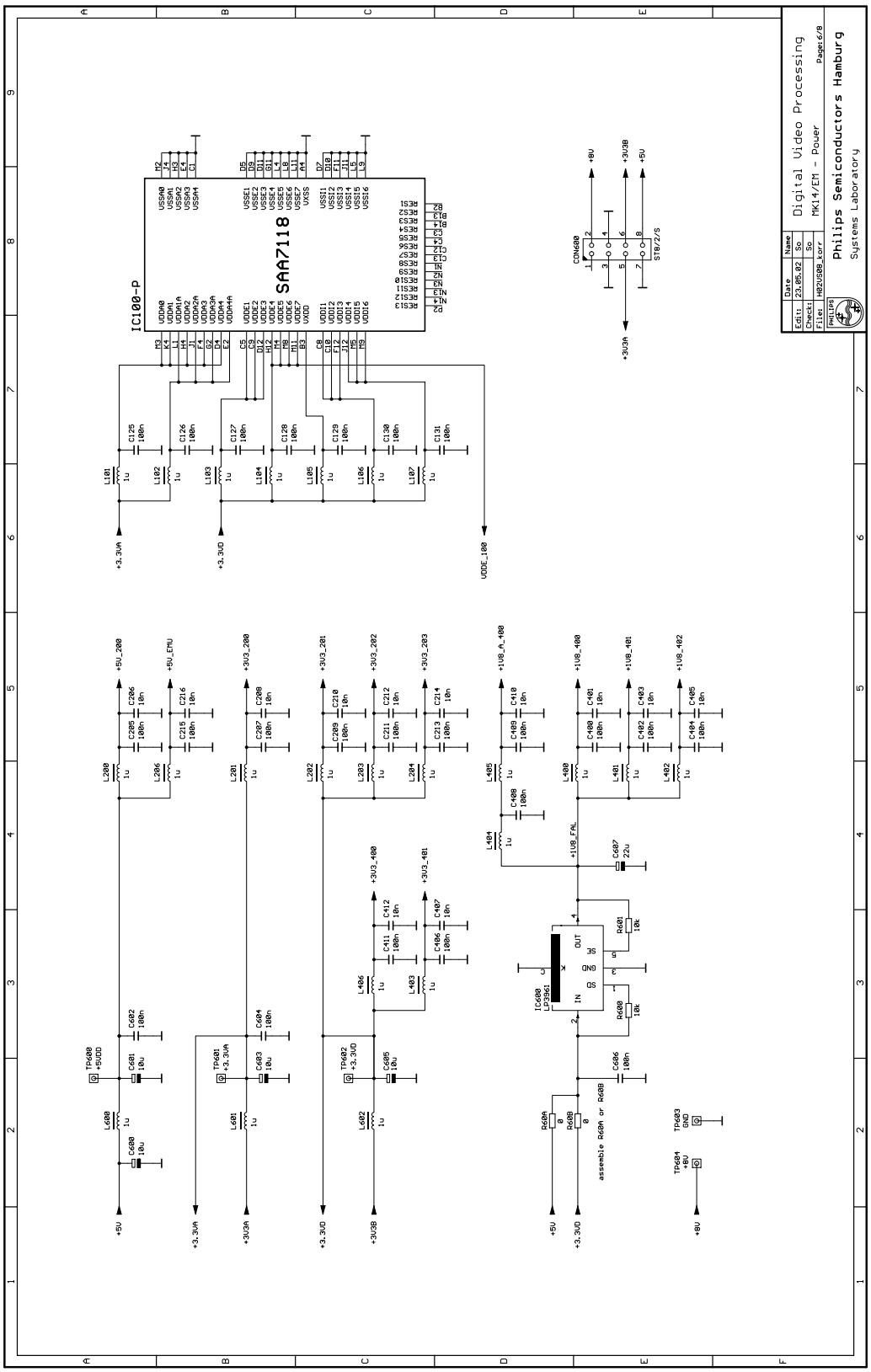


Fig. 102 IPQ module MK14-EM circuit diagram: sheet 6

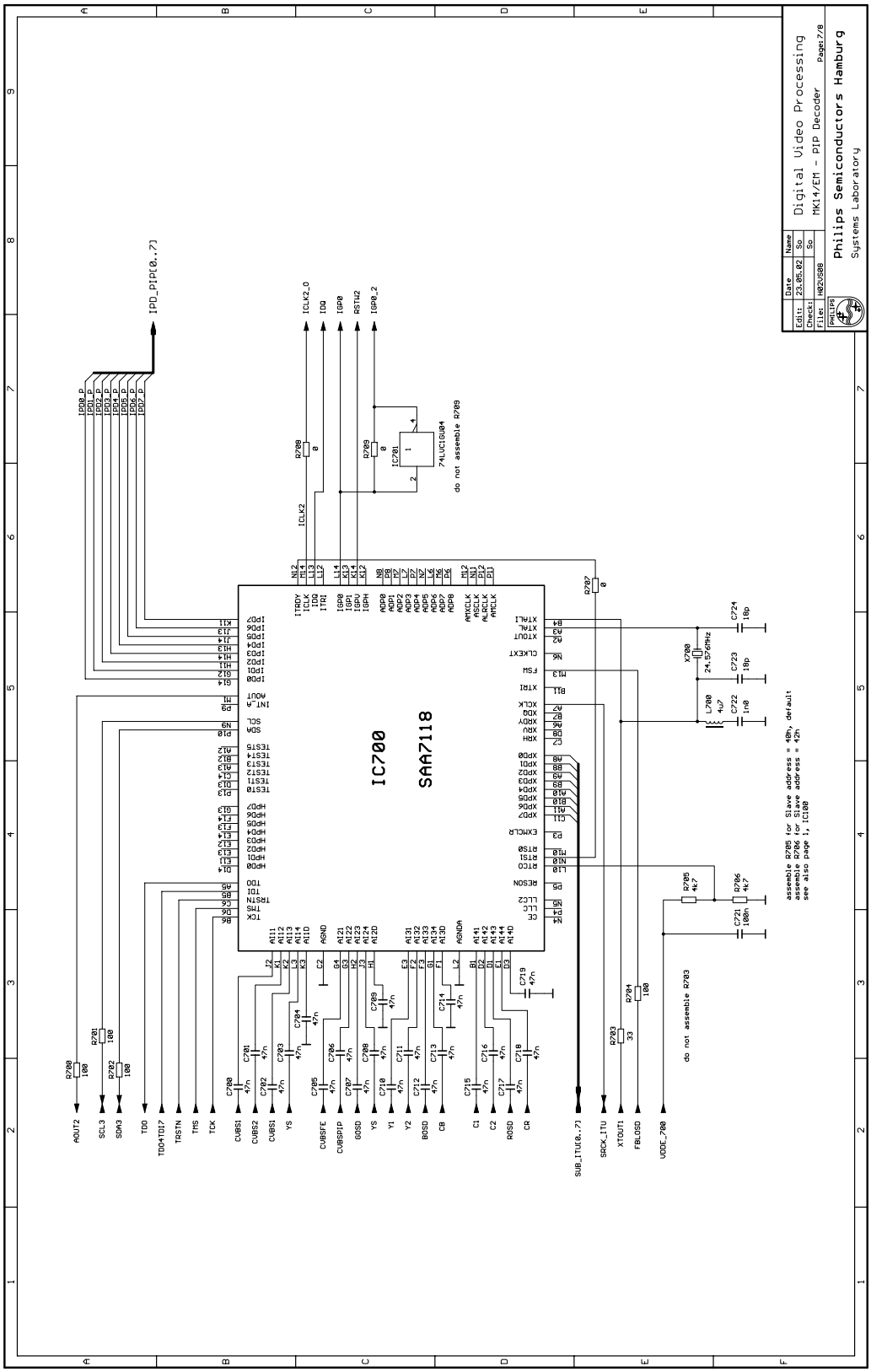


Fig. 103 IPQ module MK14-EM circuit diagram: sheet 7

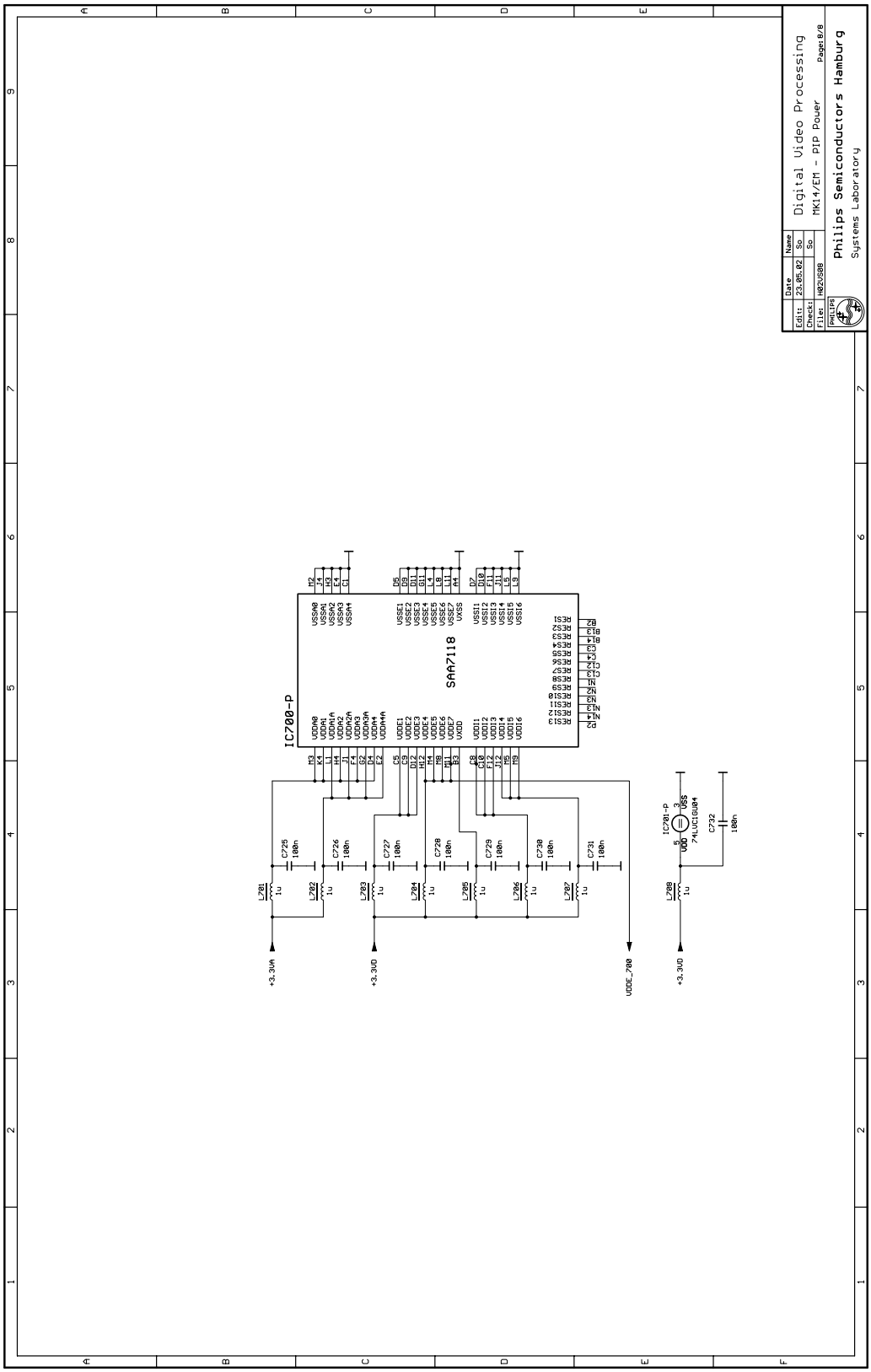


Fig. 104 IPQ module MK14-EM circuit diagram: sheet 8

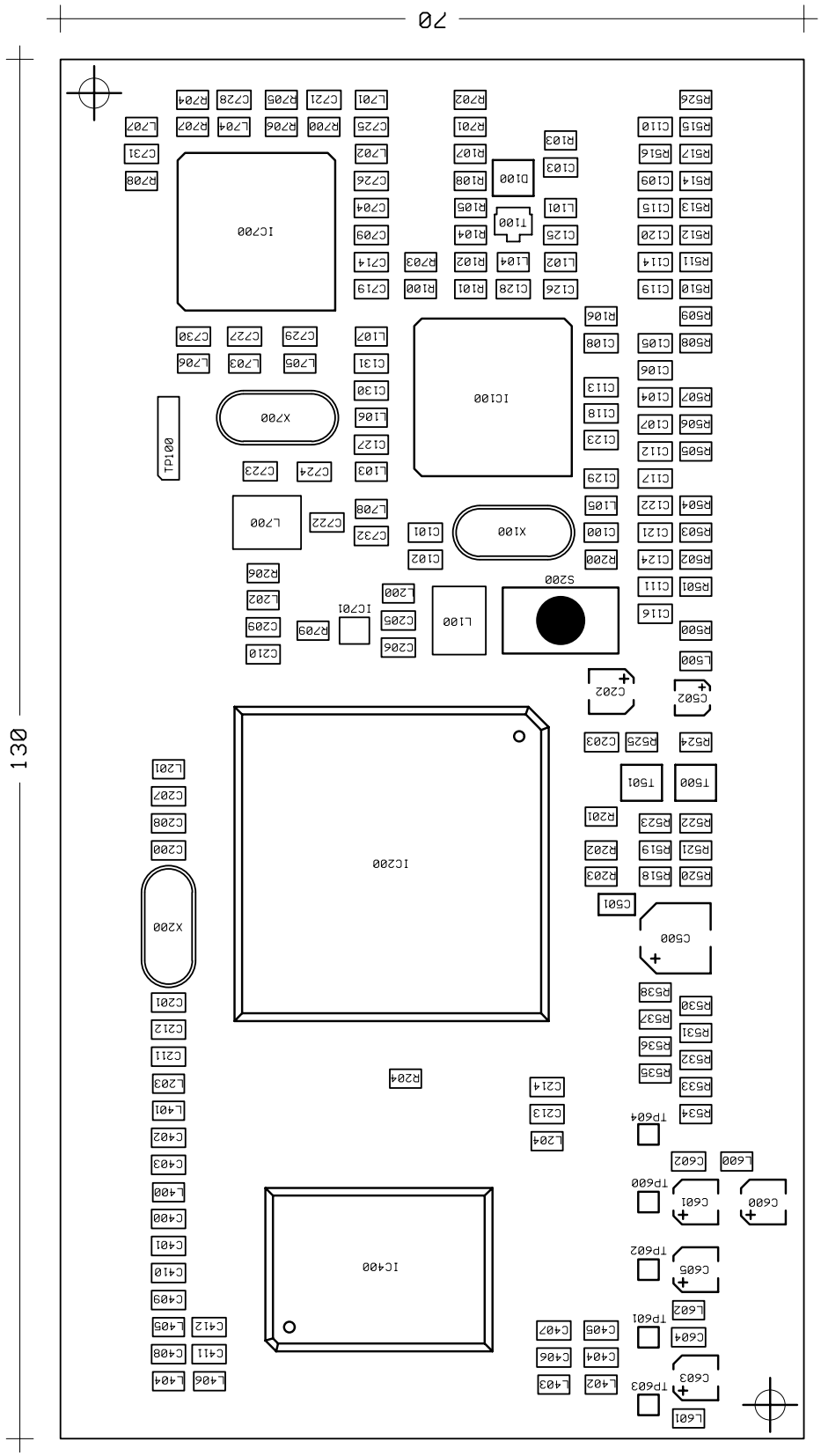


Fig. 105 IPQ module MK14-EM: position of part (top side)

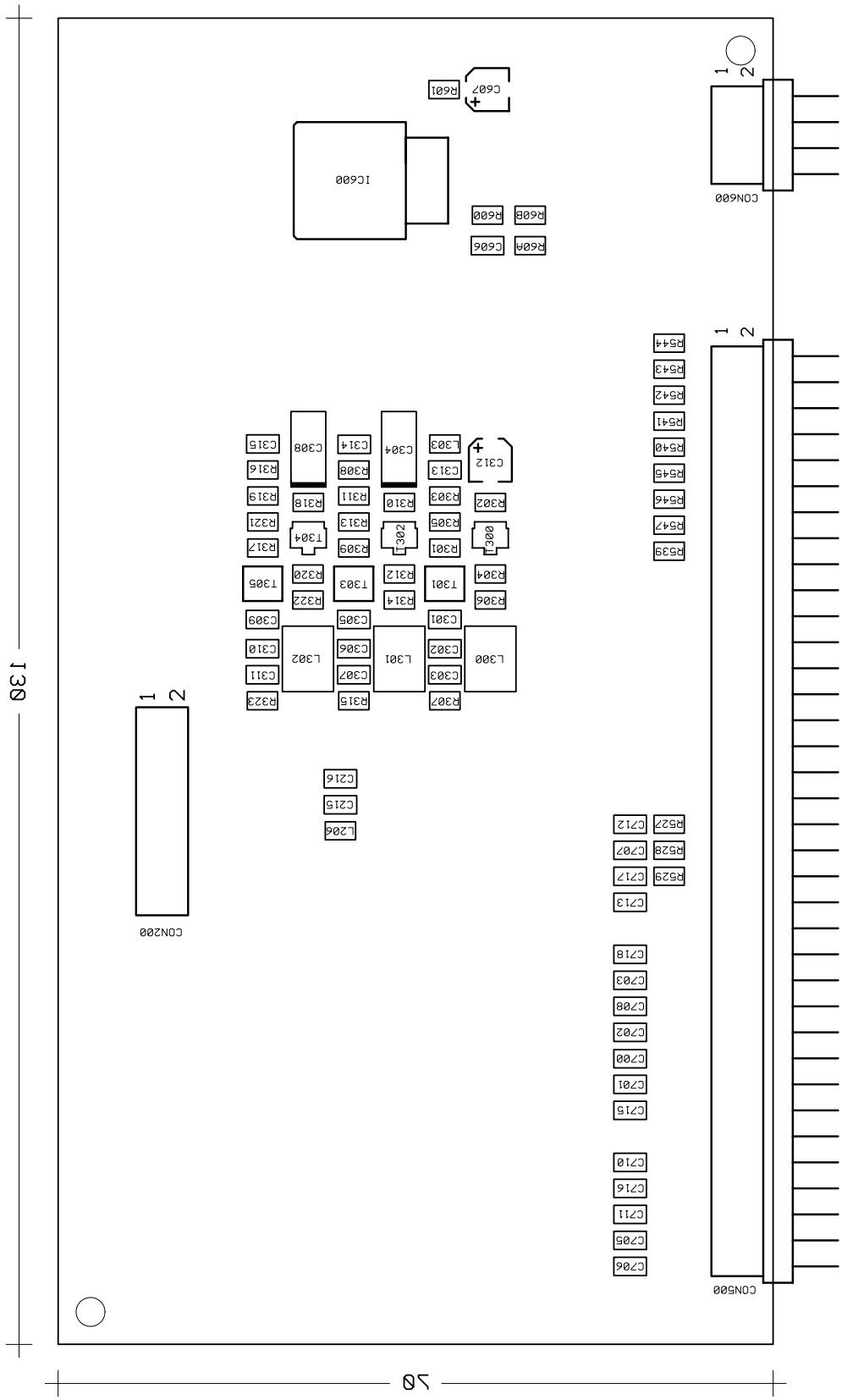


Fig. 106 IPQ module MK14-EM: position of part (bottom side)