APPLICATION NOTE

Improved Picture Quality (IPQ) Module MK12

Version: 1.0

AN01017

Philips
Semiconductors

PHILIPS

Abstract

The Improved Picture Quality (IPQ) module MK12 is a high-end scan converter module providing field rate doubling and de-interlacing (conversion to progressive scan mode). The scan converter uses the ICs SAA4992 (FALCONIC), SAA4955 and SAA4979.

The application note describes the module and its functions, gives details on circuit diagram and layout of the board and supplies a register table for control by I^2C bus.

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.

© Philips Electronics N.V. 2001

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copy-right owner.

The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or other industrial or intellectual property rights.

APPLICATION NOTE

Improved Picture Quality (IPQ) Module MK12

AN01017

Author:

Heinrich Waterholter
Systems Laboratory Hamburg,
Germany

Keywords

SAA4992
SAA4955
SAA4979
motion compensation
motion estimation
natural motion
DNR (dynamic noise reduction)

Date: May 11, 2001

Summary

The IPQ module MK12 is a high-end scan converter module providing field rate doubling and de-interlacing (conversion to progressive scan mode). The scan converter uses the ICs SAA4992 (FALCONIC), SAA4955 and SAA4979.

The conversion modes are motion compensated, this is provided by the SAA4992. This IC operates with two field memories of the SAA4955 type which provide a frame memory for image data in 4:2:2 format. A third memory for the actual scan conversion is integrated in the SAA4979.

The module is controlled by the SAA4979, a video processing IC offering video enhancing features, memory controlling and an embedded 80C51 µC core. It has two digital ITU-656 inputs and a D/A converter at the output. For communication with a master controller an I^{2}C bus interface is provided.

This application note gives an overview of the functions of the SAA4992, SAA4955 and SAA4979 and describes an application board designed to demonstrate the feature and evaluate the concept. The complete circuit diagrams are given as well as a register table for controlling the module by I^{2}C bus.

Table of Contents

Table of Figures

1. Introduction

The MK12 module is a scan converter used to convert 50 Hz or 60 Hz video input signals to 100 Hz or 120 Hz output signals. But it can be also be used as a de-interlacer to generate progressive images in 50 Hz and 60 Hz. The module offers two digital video inputs of the ITU-656 type, so two sources can be displayed at the same time, either in double window or PIP (picture-in-picture) mode. The outputs are analog YUV.

The main ICs on the board are the SAA4979, the SAA4992 and two memories SAA4955. The SAA4979 de-codes the digital video input streams, does the scan conversion together with its built-in field memory and offers various pictures improvements. After the signals have been processed, its D/A converters generate the analog YUV signals at the output. On-chip the SAA4979 also has a microcontroller with embedded RAM and ROM and an I²C interface for communication with a main controller. The SAA4992 provides motion compensation and noise reduction. The field or frame memory that the SAA4992 needs consists of either one or two SAA4955.

This application note describes the hardware functions of the SAA4979, SAA4992, SAA4955 and the application environment needed to realize 100 Hz scan conversion as well as extra functions. A register command table is added (chapter 8), 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 the MK12 IPQ module

Features mainly provided by the SAA4979 are:

- Digital YUV input according to ITU-656 standard
- 4:2:2 field rate up-conversion (50 to 100 Hz or 60 to 120 Hz)
- 3.5 MBit embedded DRAM
- Sample rate conversion for linear zoom and compression
- Panorama mode
- Dynamic noise reduction
- Noise estimator
- Black bar detection
- Luminance horizontal smart peaking
- Digital Color Transient Improvement (DCTI)
- Triple 10-bit Digital-to-Analog Converter (DAC)
- Line locked PLL
- Expansion port for SAA4992H/SAA4991WP
- Double window and picture-in-picture processing
- Embedded 80C51 microprocessor
- 32 KByte internal ROM (mask programmable)
- 512 byte internal RAM
- I^{2}C-bus controlled
- Synchronous No parity Eight bit Reception and Transmission (SNERT) interface

Features of the SAA4992 are:

- Upconversion of all 1f_H film and video standards up to 292 active input lines per field
- 100/120 Hz 2:1, 50/60 Hz 1:1 and 100/120 Hz 1:1 output formats
- 4:1:1, 4:2:2 and 4:2:2 DPCM input color formats; 4:1:1 and 4:2:2 output color formats
- Full 8-bit accuracy
- Scalable performance by applying 2 or 3 external field memories
- Improved recursive de-interlacing
- Film (25 Hz, 30 Hz) upconversion to 100/120 movement phases per second
- Variable vertical sharpness enhancement
- Motion compensated 3D dynamic noise reduction
- High quality vertical zoom

3. Block diagram of the MK12 IPQ module

Fig. 1 shows the block diagram of the MK12 scan converter module. Its main ICs are the SAA4979 which does the actual scan conversion, and the SAA4992 which provides motion estimation and compensation. For full performance the SAA4992 needs two field memories of the SAA4955 type. The conversion can either be scan rate doubling (50 / 60 Hz input converted to 100 / 120 Hz output) or de-interlacing (progressive scan mode, double line frequency).

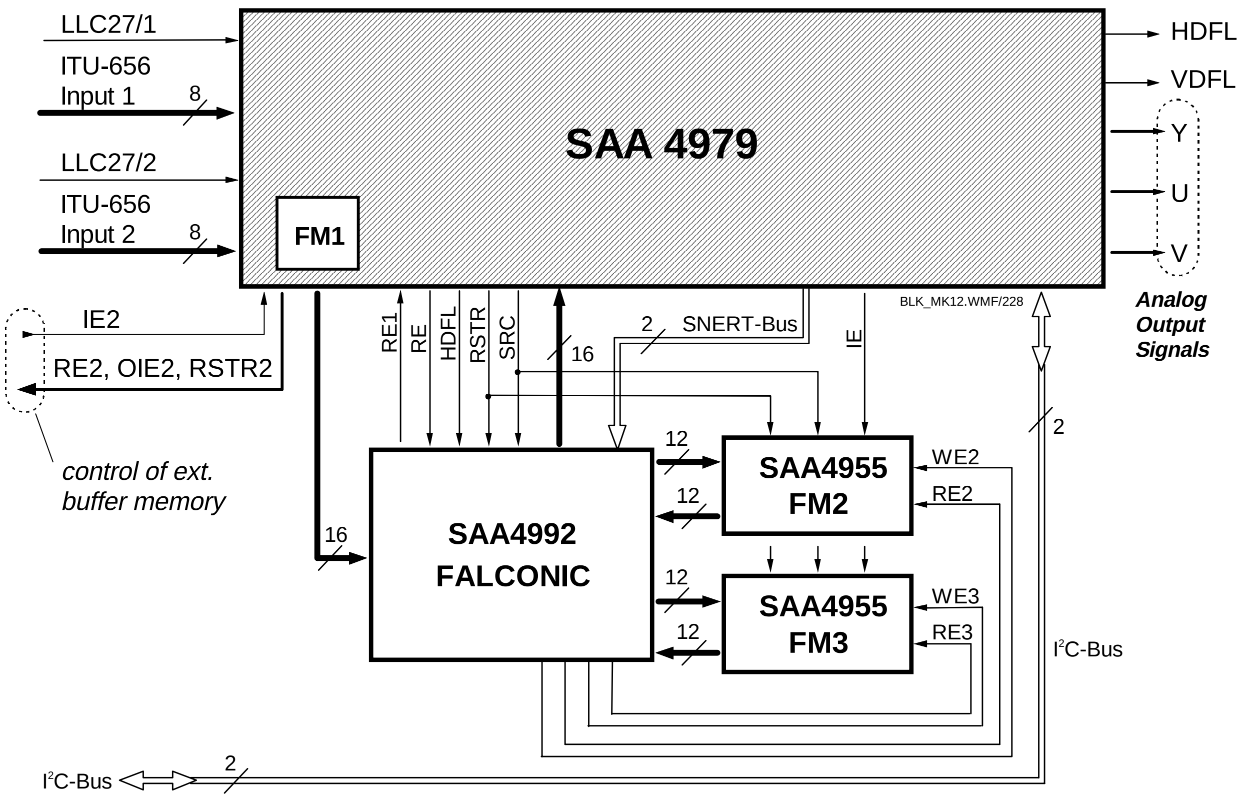


Fig. 1 Block diagram of the MK12 IPQ module

The SAA4979 has two inputs for digital video data, a main one (main channel, input 1) and a secondary one (subchannel, input 2). Static switching between these inputs is possible, e. g. for source selection, but also fast switching is provided in order to display both inputs at the same time. E. g. this could be a double-window (DW) or picture-in-picture (PIP) display. For simultaneous display of two video sources these must be synchronized in horizontal and vertical timing. To achieve this a buffer memory is needed in front of the second input. The control signals to operate such a buffer memory are provided by the SAA4979.

At the output the SAA4979 provides digital-to-analog (D/A) converters which output the luminance and chrominance components Y, U and V for easy connection to a conventional video processor. Appropriate horizontal (HDFL) and vertical (VDFL) synchronization signals are also provided.

The SAA4979 has a built-in memory for scan conversion purposes. Therefore a 100 Hz scan converter for simple field repetition can be built with this IC alone (one-chip 100 Hz). Easy upgrading then is possible by adding a motion compensation IC with one or two additional field memories.

The SAA4979 uses a built-in memory controller to write and read data to and from the internal scan conversion memory. This controller also provides signals to operate the motion compensation IC and its memories as well as the buffer memory at the subchannel input.

The SAA4979 has an internal 8051 microprocessor core with RAM and ROM. This microprocessor controls the internal functional blocks, the internal memory controller as well as the SAA4992 which is connected to the SAA4979 by a serial two-wire bus (SNERT[1]-bus).

The SAA4992 is a motion compensation IC which is connected to the expansion port of the SAA4979. It has one or two field memories attached which serve as basis for the motion estimation algorithm. For full performance, expecially in movie processing, two field memories are necessary.

Throughout the module video signals in the 4:2:2 format are supported.

1. SNERT stands for *Synchronous No-parity Eight bit Reception and Transmission*

4. Functional description of the SAA4979

The SAA4979 is a stand-alone IC for 4:2:2 video scan conversion from 50/60 Hz to 100/120 Hz. 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 one-chip 100 Hz conversion. It is the successor of the SAA4977. For mid- and high-end applications it also offers an expansion port for vector based motion estimation and compensation ICs like the SAA4991WP (MELZONIC) or SAA4992H (FALCONIC). Among others, FALCONIC provides features like motion-compensated field rate up-conversion, de-interlacing, noise reduction and zoom.

The functional block diagram of the SAA4979 is shown in fig. 2. 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. 3. 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. 4 where the position of horizontal timing reference codes is depicted.

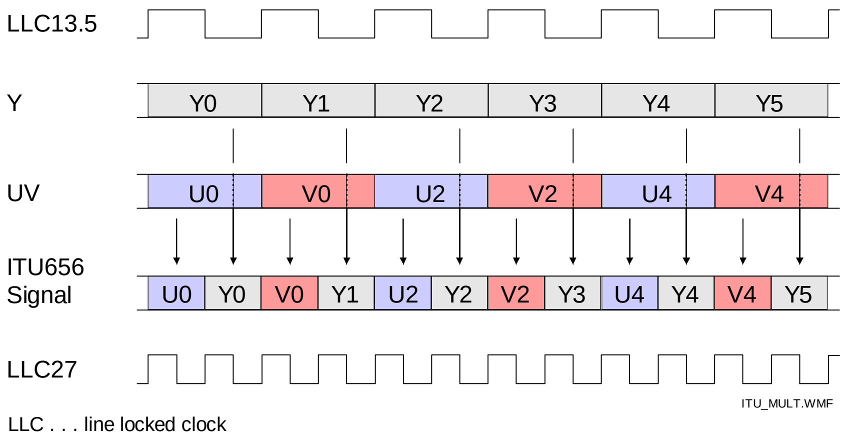


Fig. 3 ITU-656 multiplex signal

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. 5.

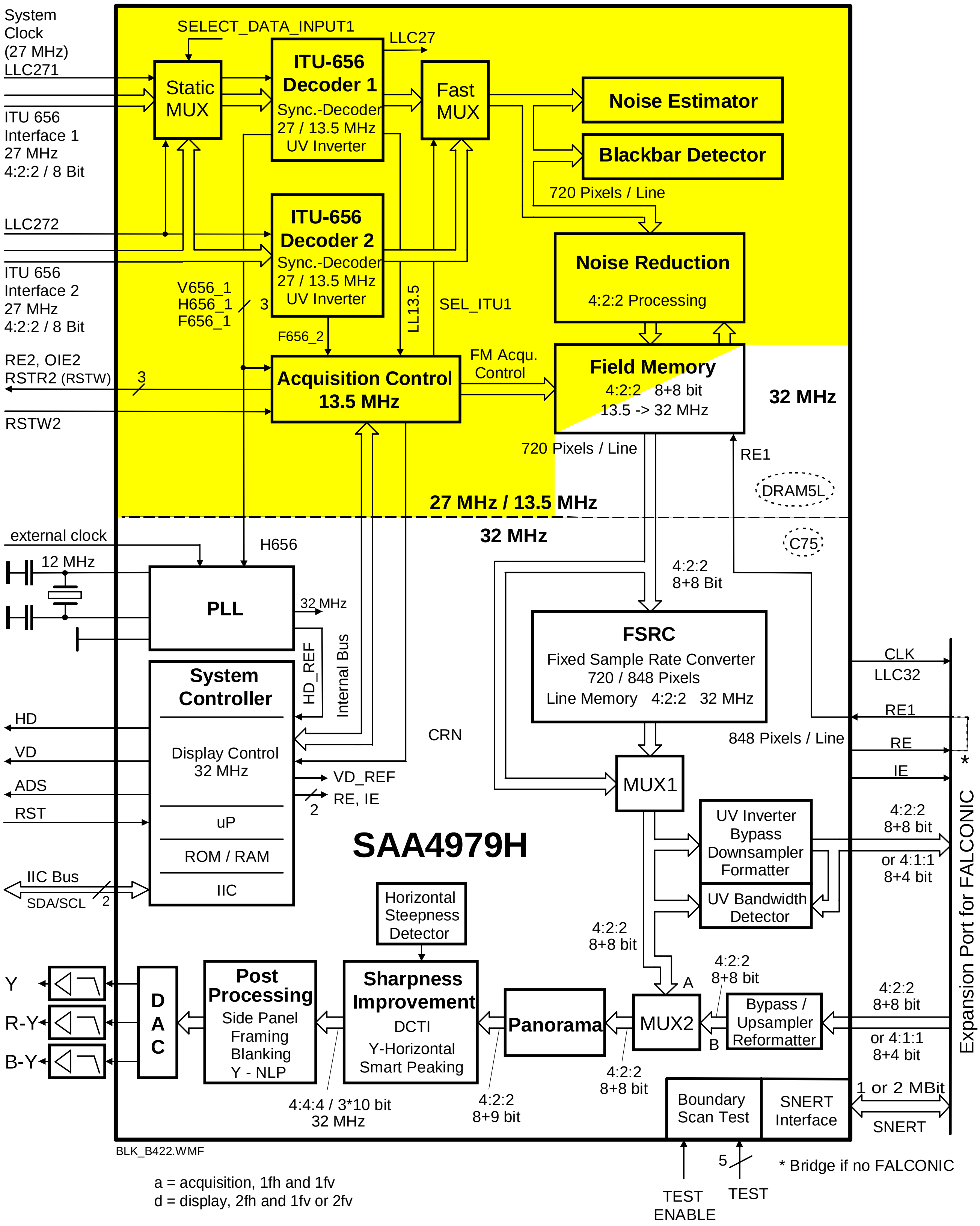
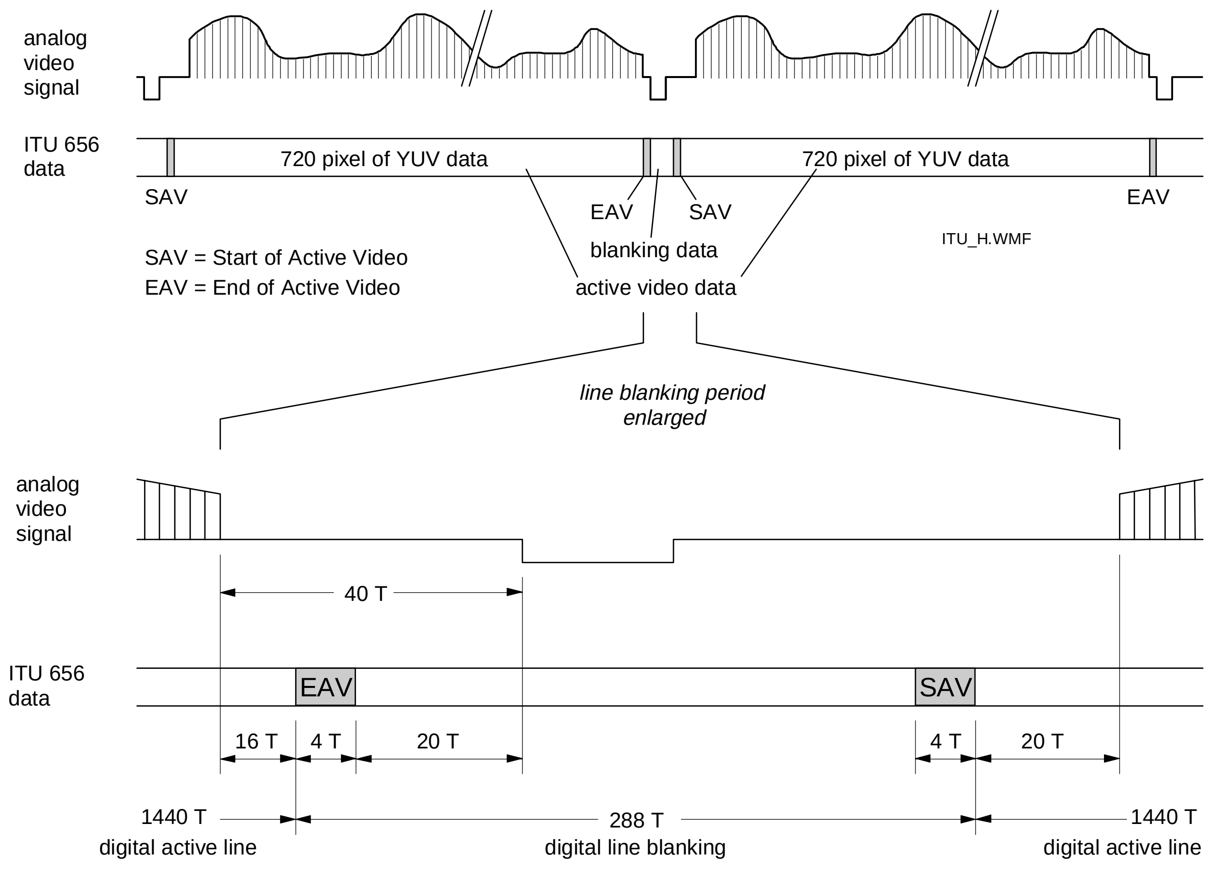


Fig. 2 Block diagram of the SAA4979H



LLC27 = 27 MHz
T = 1/LLC27 = 37 ns
digital line = digital line blanking + digial active video = 1728T

Fig. 4 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. 5 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. 6.

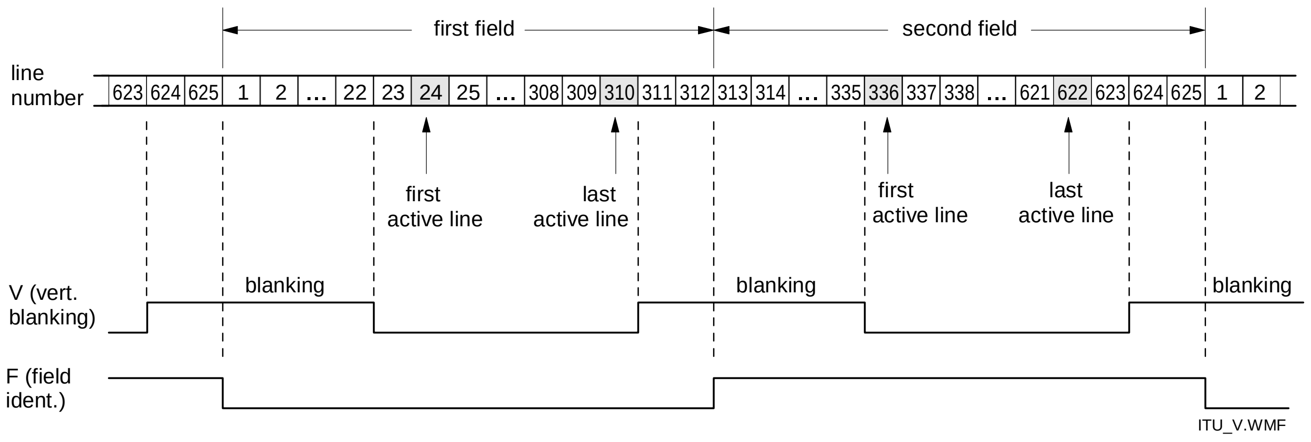


Fig. 6 ITU-656 vertical timing

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.

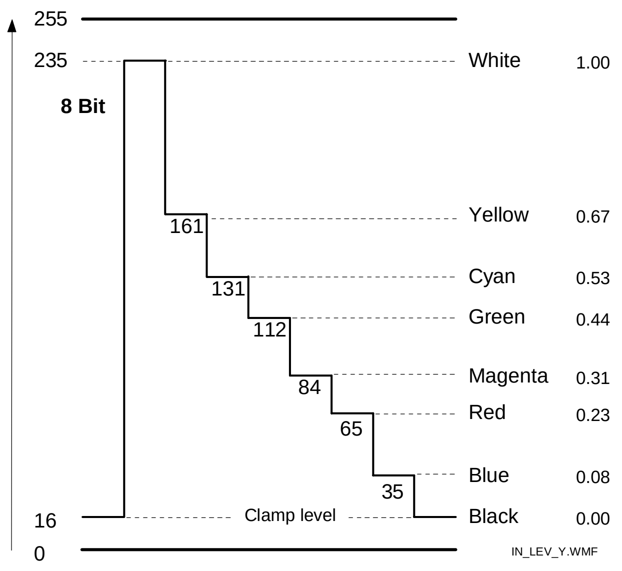


Fig. 7 Digital levels of Y input signal for color bar 100/0/75/0 (ITU-601)

The levels of the input signal components Y, Cr, Cb (”YUV”) are depicted in fig. 7, 8, and 9, the right most scale is normalized to the amplitude 1.

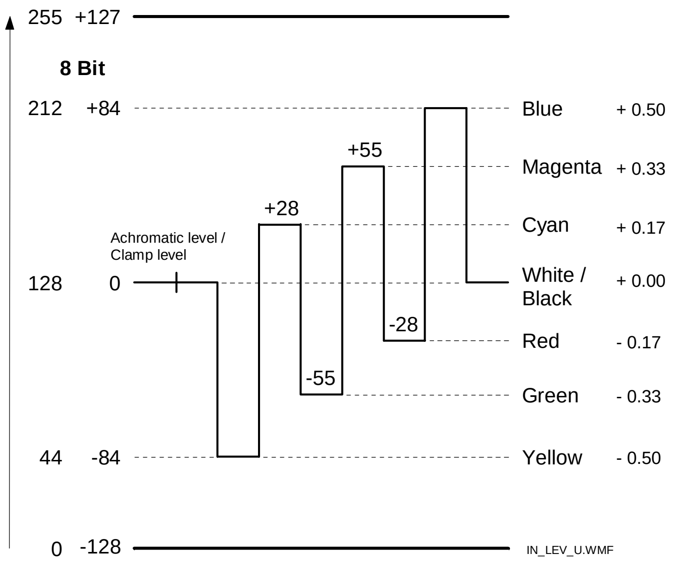


Fig. 8 Digital levels of U input signal for color bar 100/0/75/0 (ITU-601)

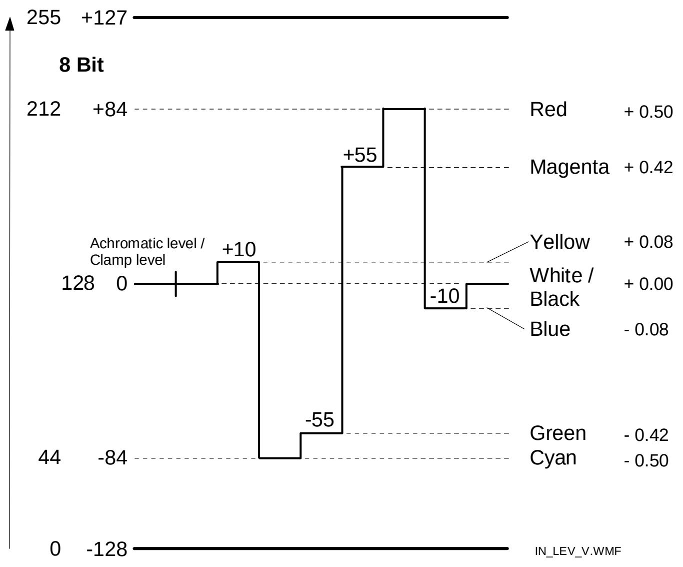


Fig. 9 Digital levels of V input signal for color bar 100/0/75/0 (ITU-601)

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. 10. If no measures are taken, they will also leave these black bars on a wide

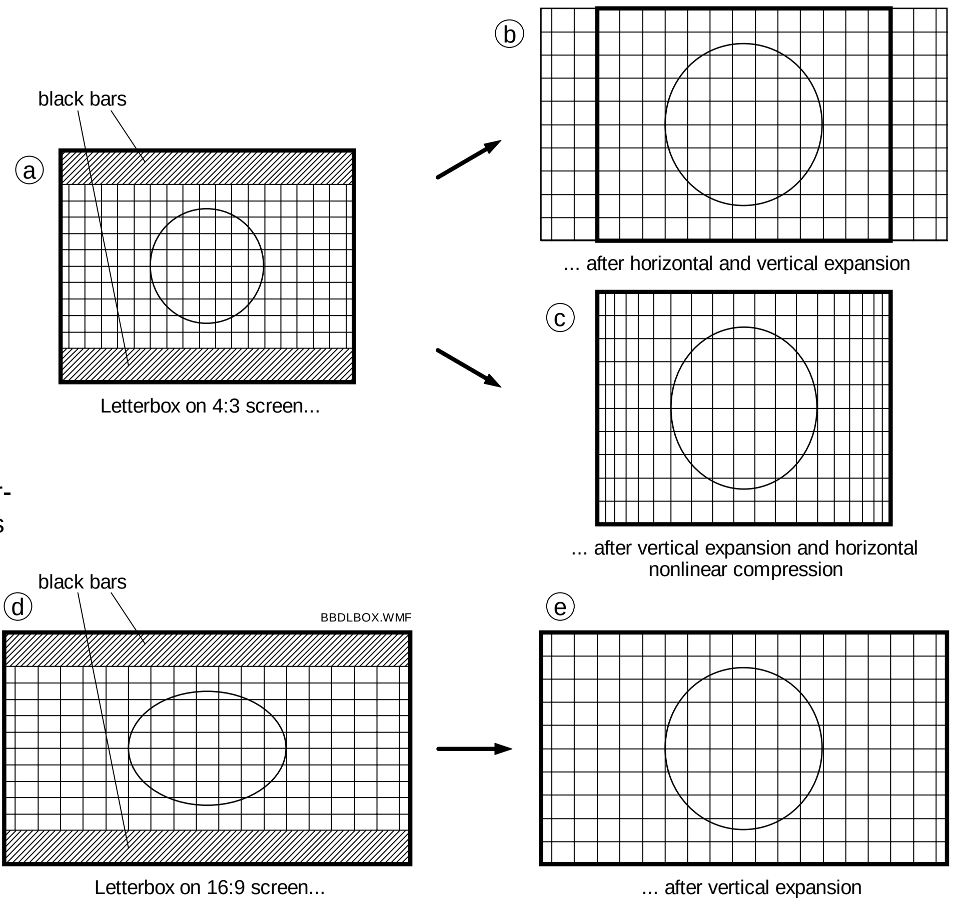

black bars
ⓐ
Letterbox on 4:3 screen...
ⓑ
... after horizontal and vertical expansion
ⓒ
... after vertical expansion and horizontal nonlinear compression
black bars
ⓓ
BBDLBOX.WMF
Letterbox on 16:9 screen...
ⓔ
... after vertical expansion

Fig. 10 Dealing with letterbox transmissions

16:9 screen, see ⓓ in fig. 10. 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. 10 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. 10). 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. 10).

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. 10).

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. 11 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.

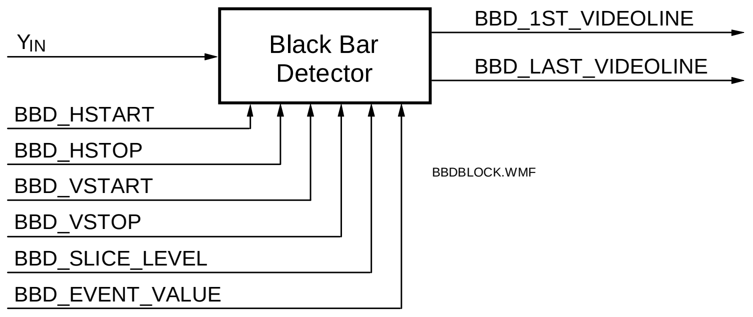


Fig. 11 Block diagram of the black bar detection

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. 12 shows the block diagram of the noise reduction circuit. A more detailed diagram can be found in the data sheet of the SAA4979[1].

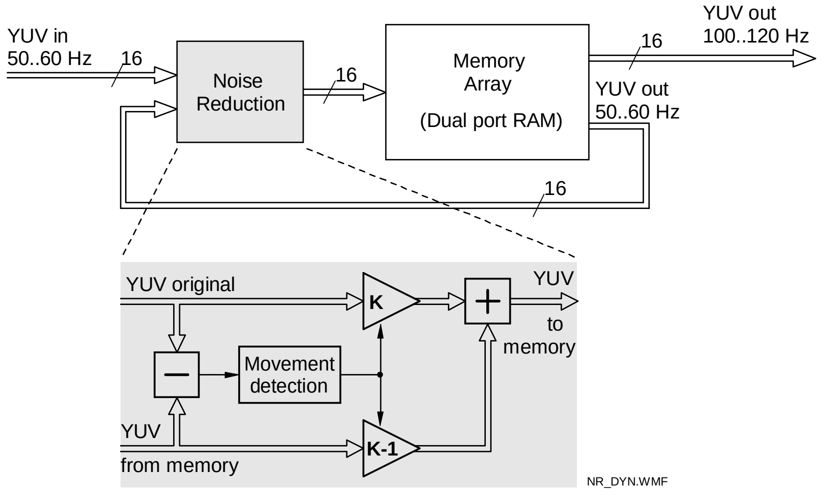


Fig. 12 Basic block diagram of the DNR circuit

1. SAA4979H: Philips Semiconductors data sheet

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 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. 13 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. 13 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. 14.

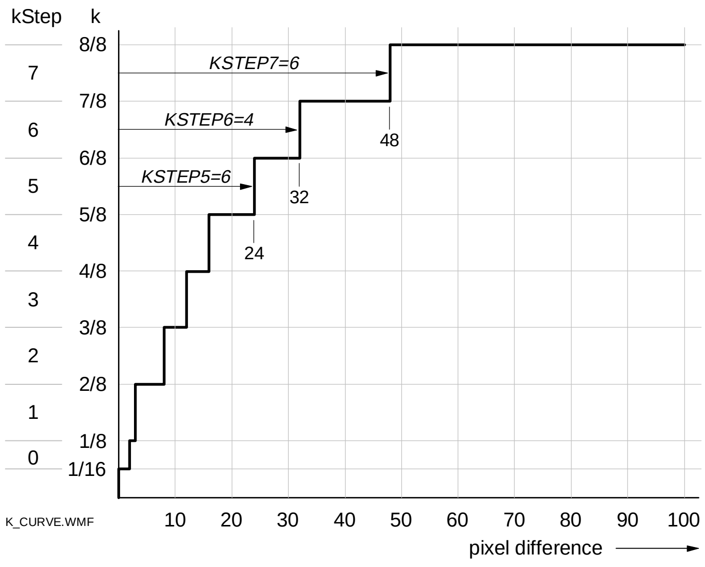


Fig. 13 Sample noise reduction k-curve

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. 14 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).

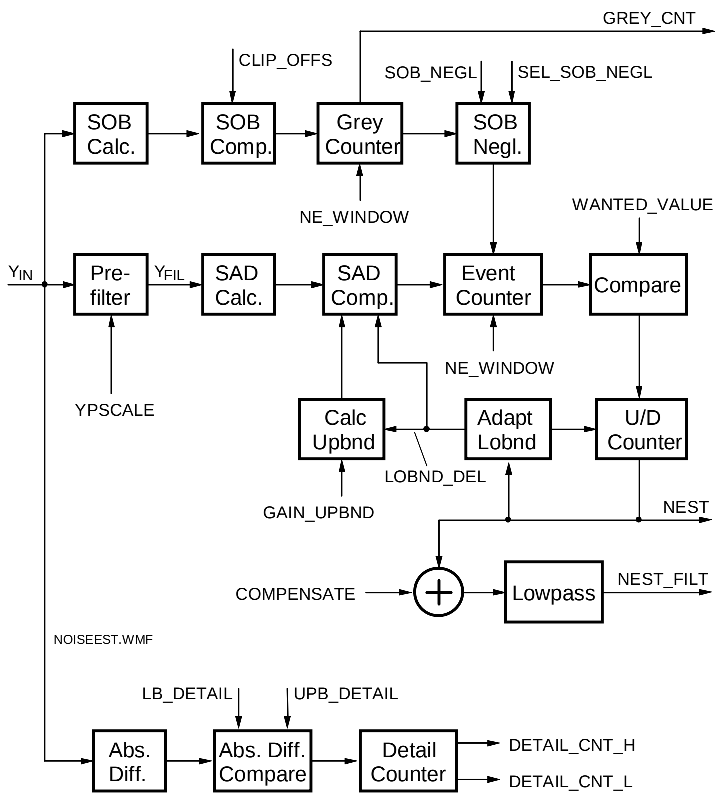


Fig. 15 Block diagram of noise estimator

A block diagram of the noise estimator is given in fig. 15. 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. 16. 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. 16 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. 17.

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. 17 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. 12. 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 pulse which is derived from the 50 Hz vertical sync pulse V656_1, the read address pointer is reset by the RSTR pulse which occurs in the vicinity of the vertical 100 Hz sync pulse VD. Whenever WE is active, data is written to the memory, and whenever RE is active, data is read from the memory. WE and RE are generated by the acquisition control part within the SAA4979. Fig. 18 shows the basic timing of the scan conversion memory. (RSTW, V656_1, WE and RE are chip internal signals).

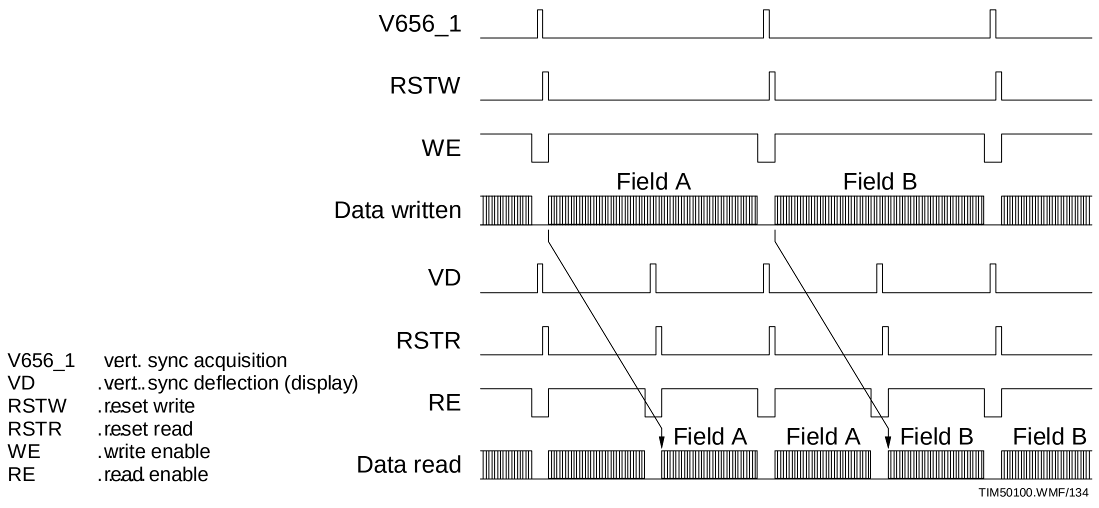


Fig. 18 Basic timing of the scan conversion memory

The RSTW 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*).

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).

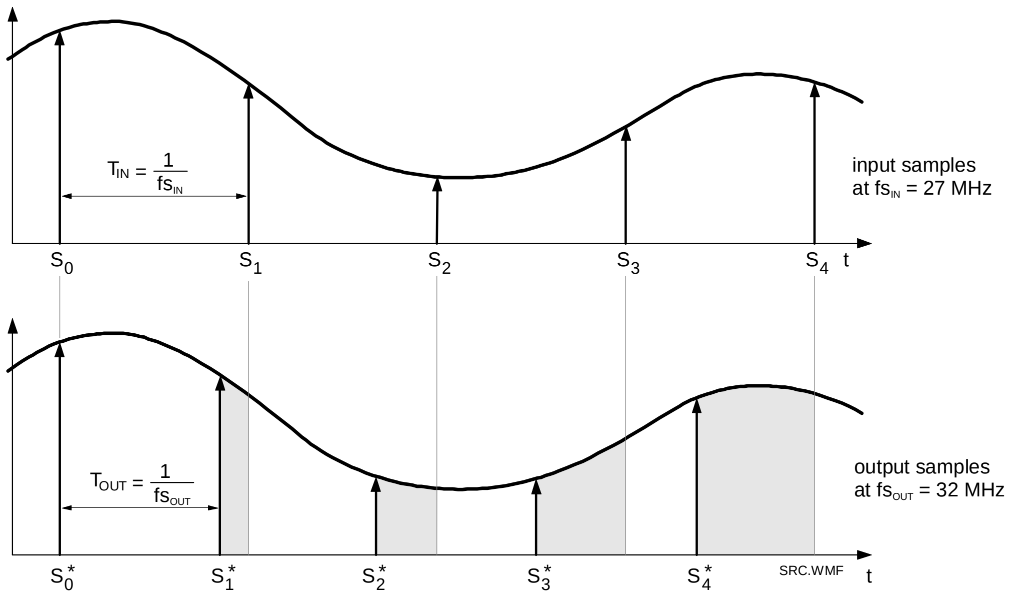


Fig. 19 Sample rate conversion by interpolation

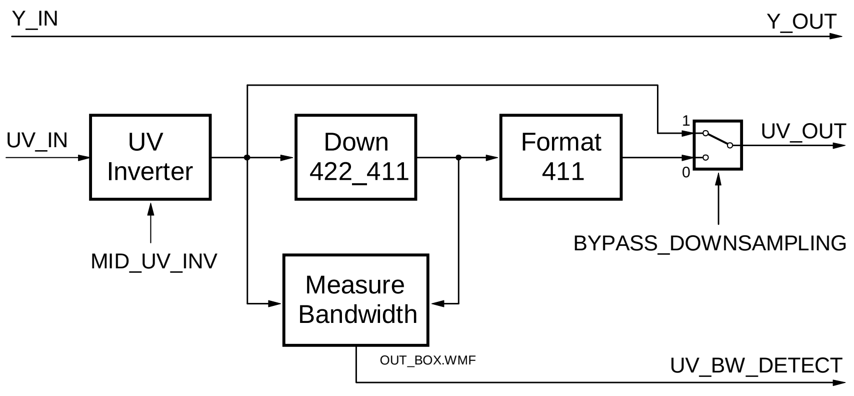


Fig. 20 Block diagram of the output part of the expansion port

Fig. 20 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 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. 21 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

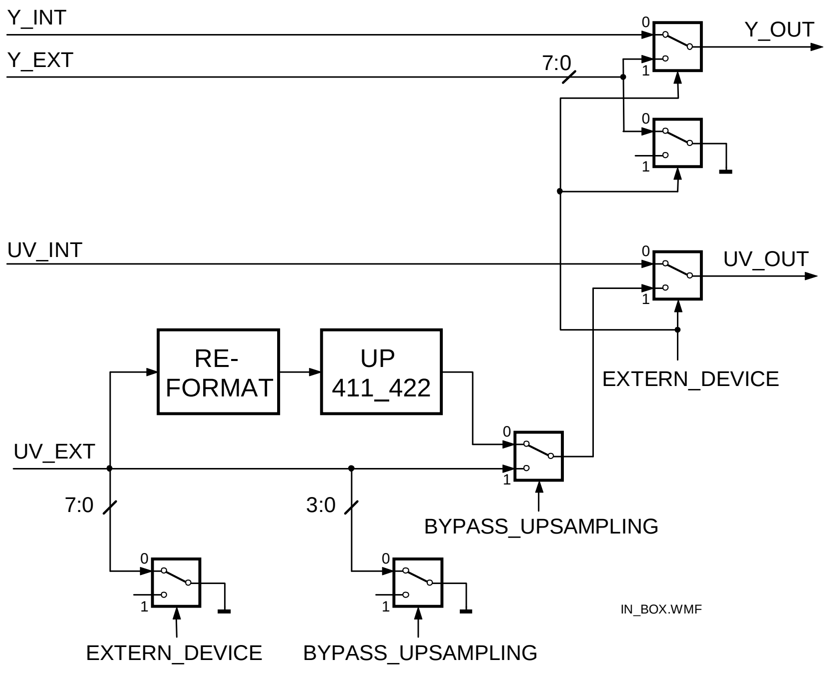


Fig. 21 Block diagram of the input part of the expansion port

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:

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

- 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

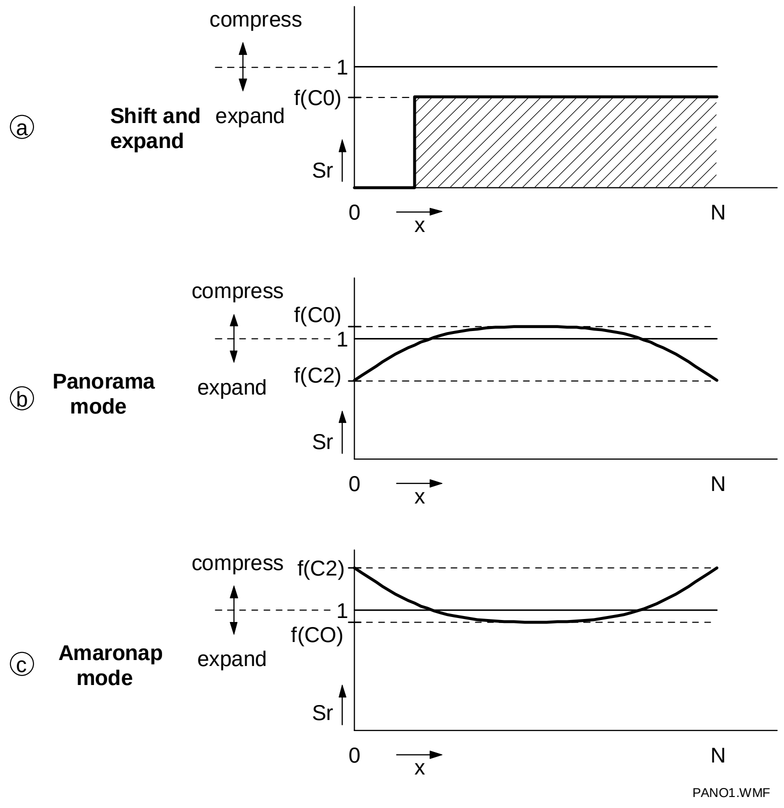
  With a zero order variation of the delay a linear compress or expand function is obtained. The range for the compression factor is 0 to 2, meaning infinite zoom to a compression factor of 2. The amount of compression or expansion is determined by the parameter *C0* (zero order control). In fig. 22 a 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. 22b). 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. 23).

  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. 22 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.



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

Fig. 22 Principle of panoramic zoom

Fig. 23 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.

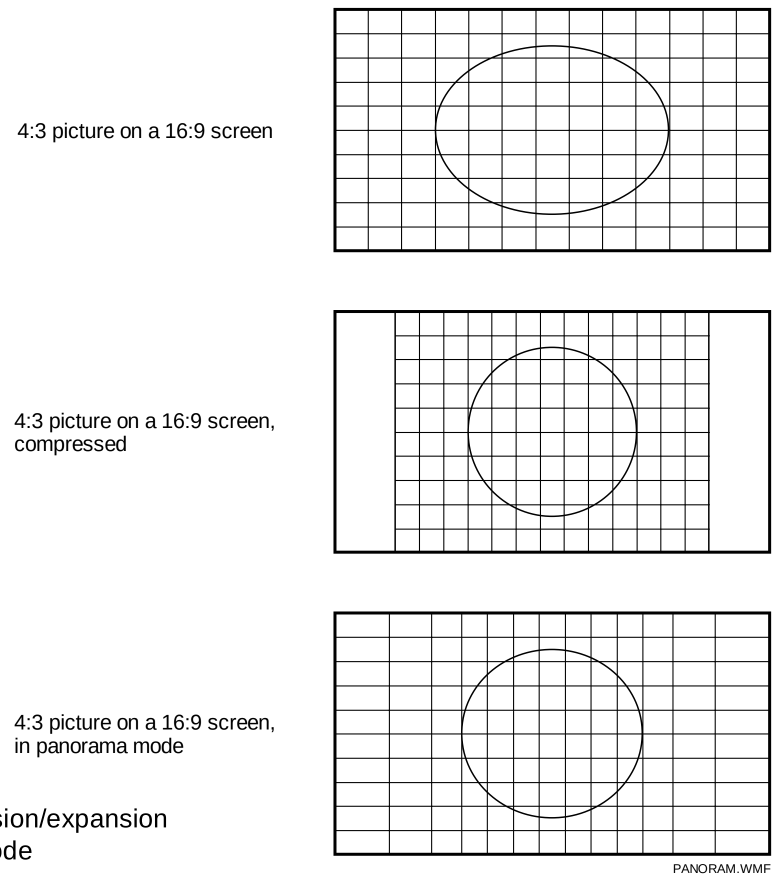

4:3 picture on a 16:9 screen
4:3 picture on a 16:9 screen, compressed
4:3 picture on a 16:9 screen, in panorama mode
PANORAM.WMF

Fig. 23 Nonlinear compression/expansion in panorama mode

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. 24. U and V data always 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.

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. 25. 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.

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

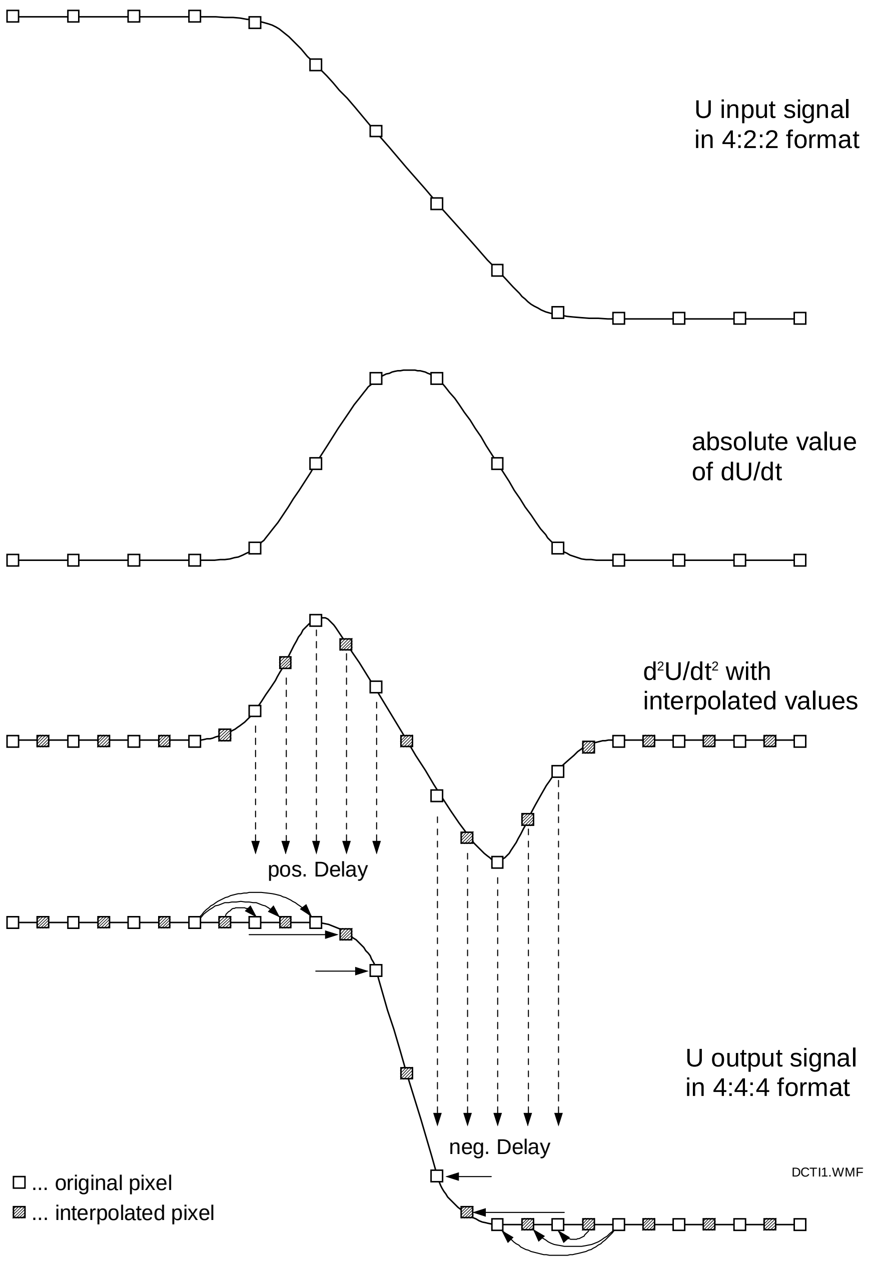
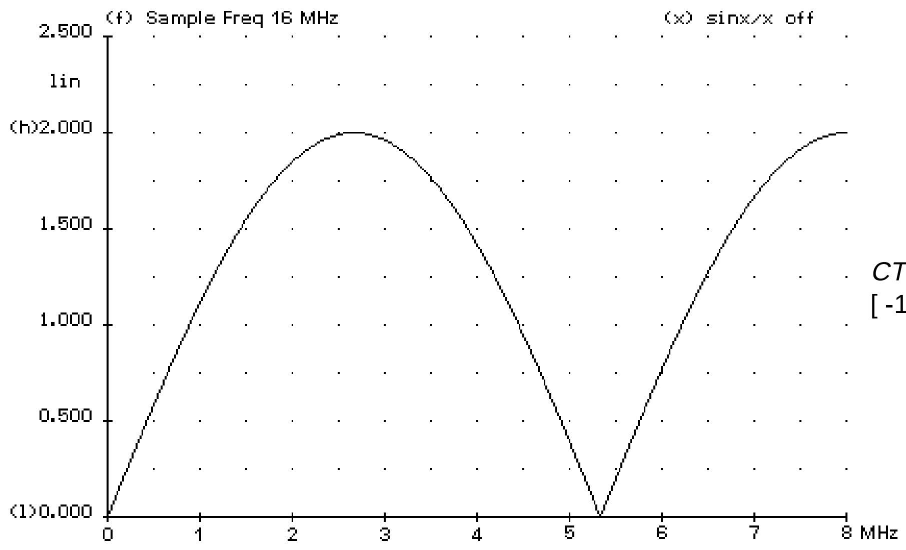


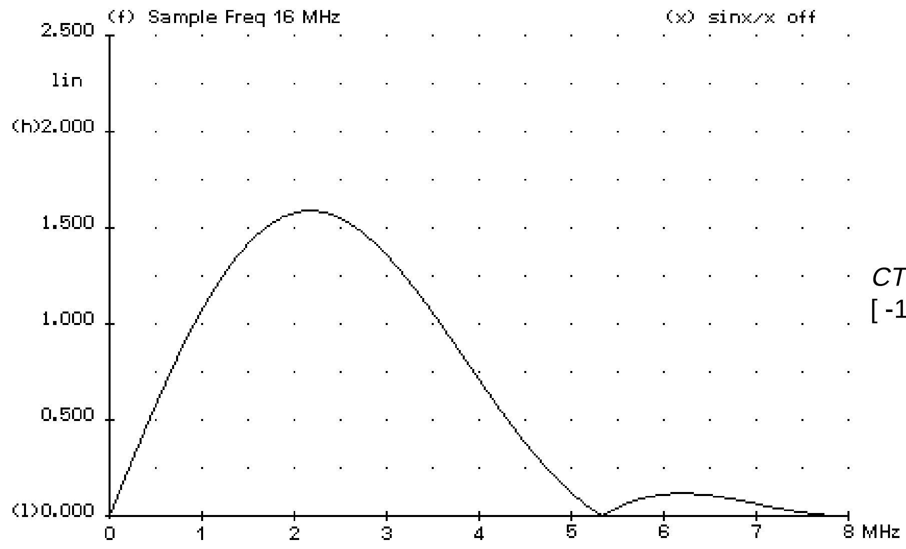
Fig. 24 DCTI basic operating principle

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. 26 using a maximum amplitude color transient as input signal.



CTI_DDX_SEL = 0:
[-1 0 0 -1]



CTI_DDX_SEL = 1:
[-1 -2 -1 1 2 1]

Fig. 25 Transfer curves of the first differentiating filter

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. 27 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.

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. 29 and 30 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. 28. 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*.

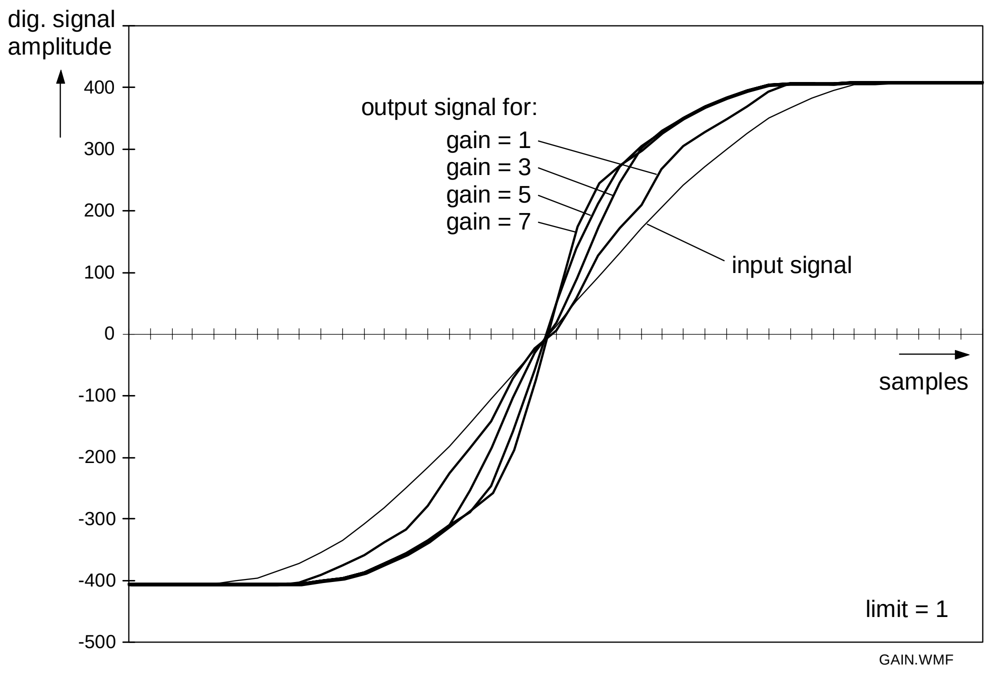


Fig. 26 DCTI with variation of gain for a limit setting of 1

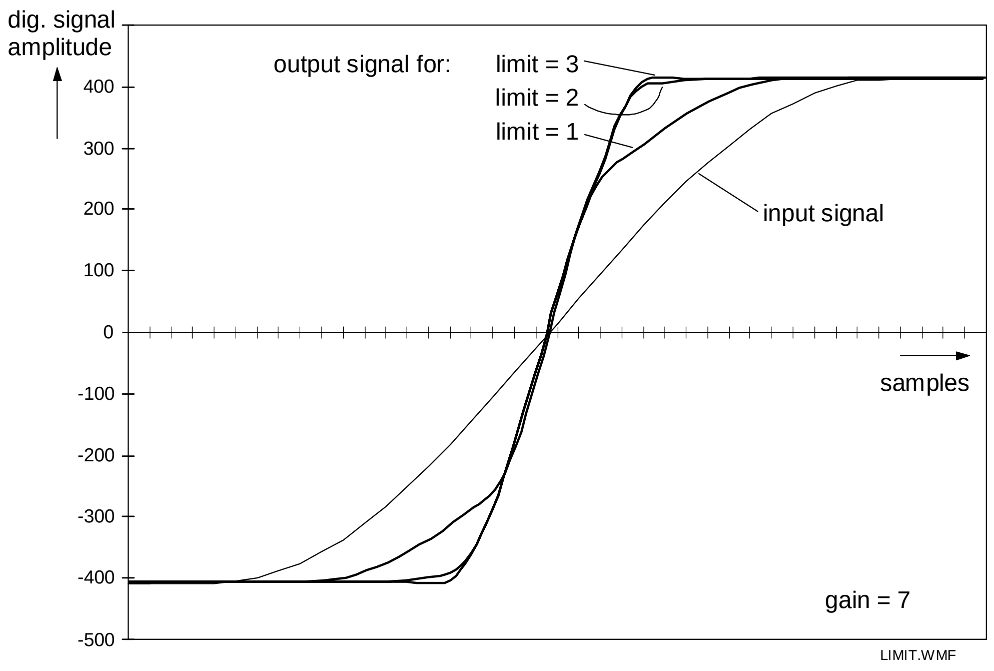


Fig. 27 DCTI with variation of limit for a gain setting of 7

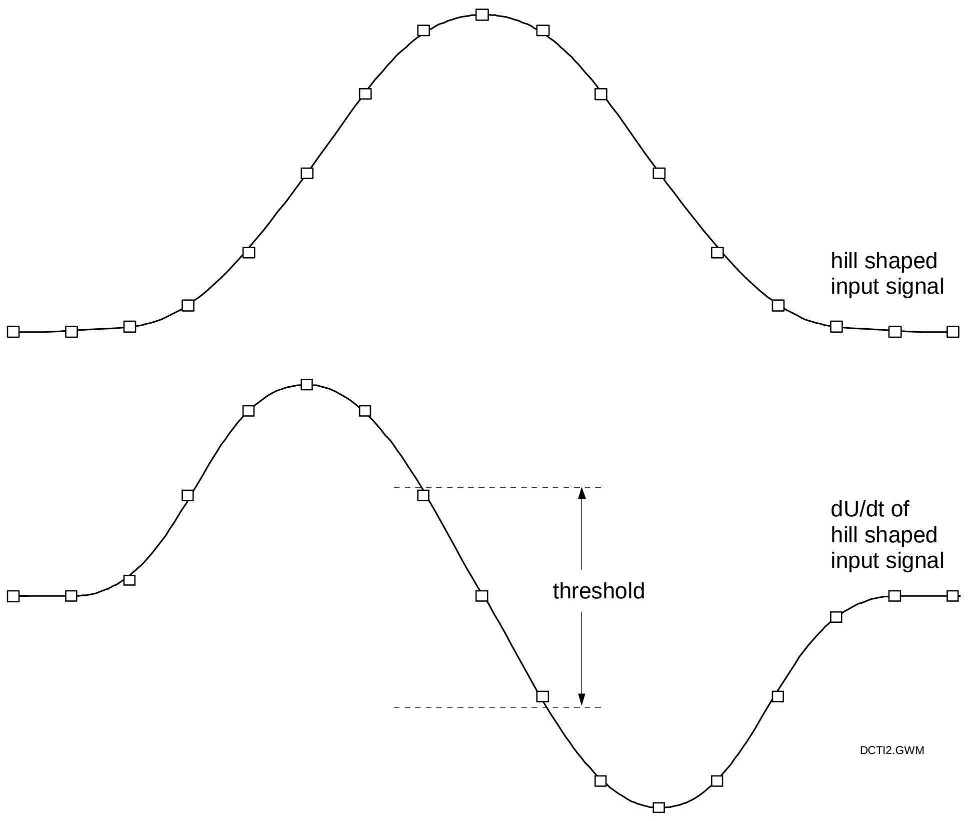


Fig. 28 Principle of hill detection

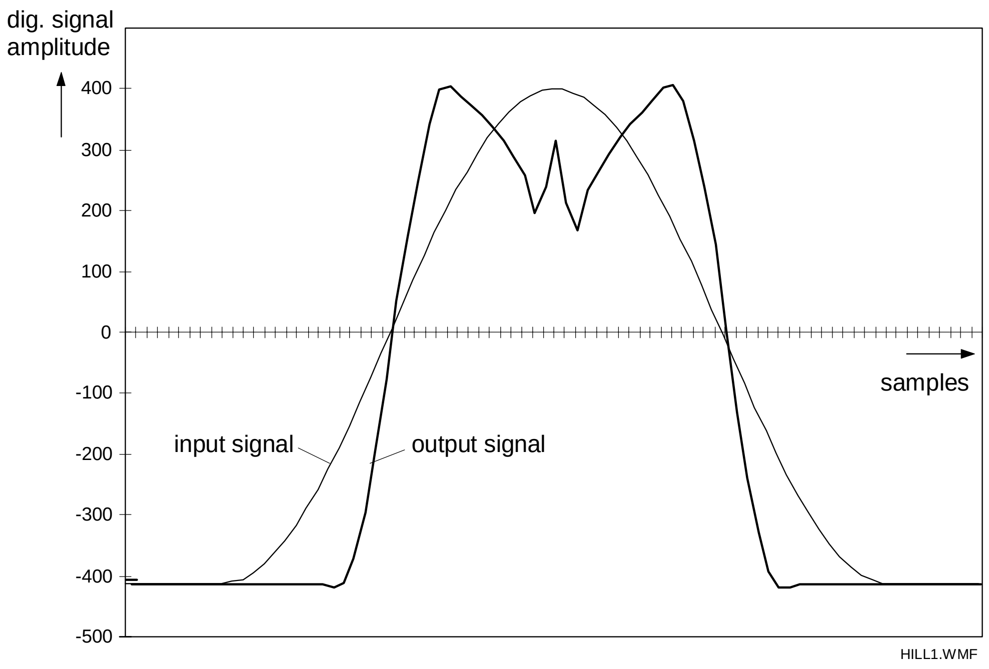


Fig. 29 DCTI without ´over-the-hill protection´

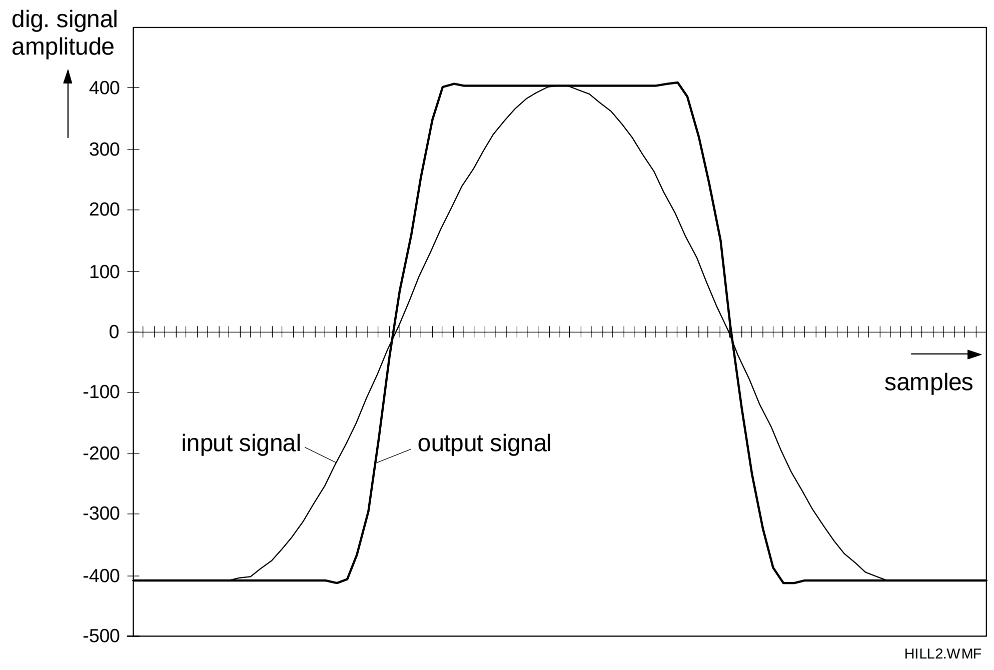


Fig. 30 DCTI with over-the-hill-protection

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. 31 and 32. 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. 33. 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.

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 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. 34 and 35.

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. 36. 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

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

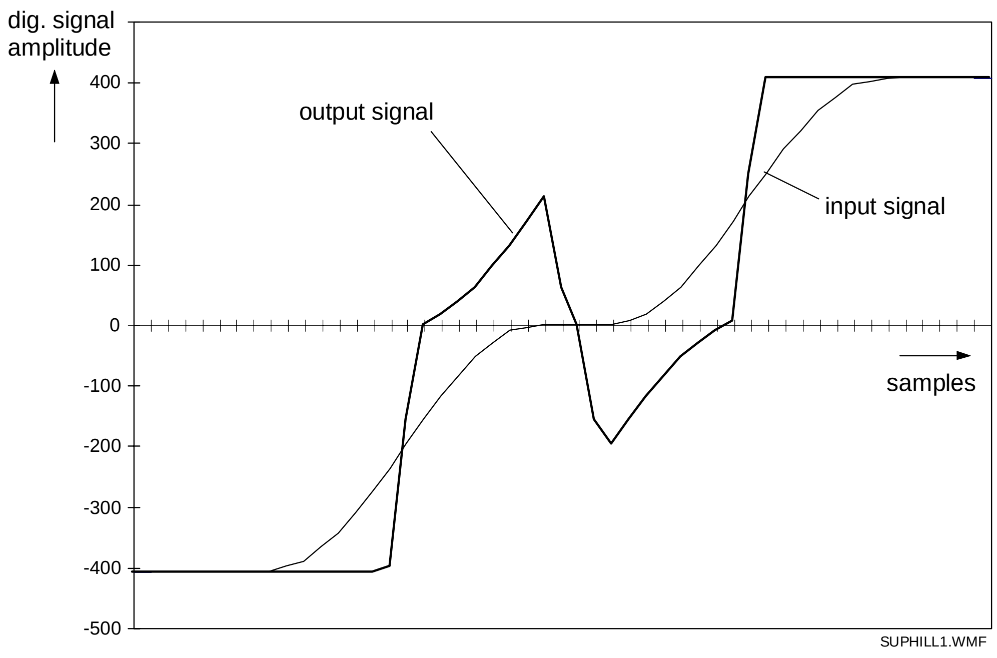


Fig. 31 DCTI with superhill-protection off

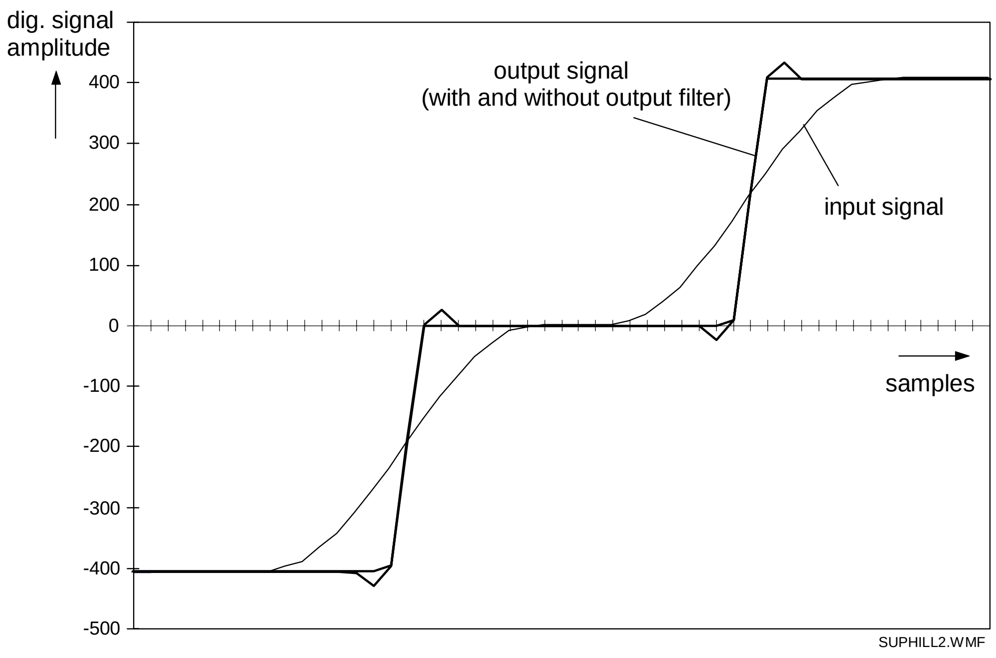


Fig. 32 DCTI with superhill-protection on

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.

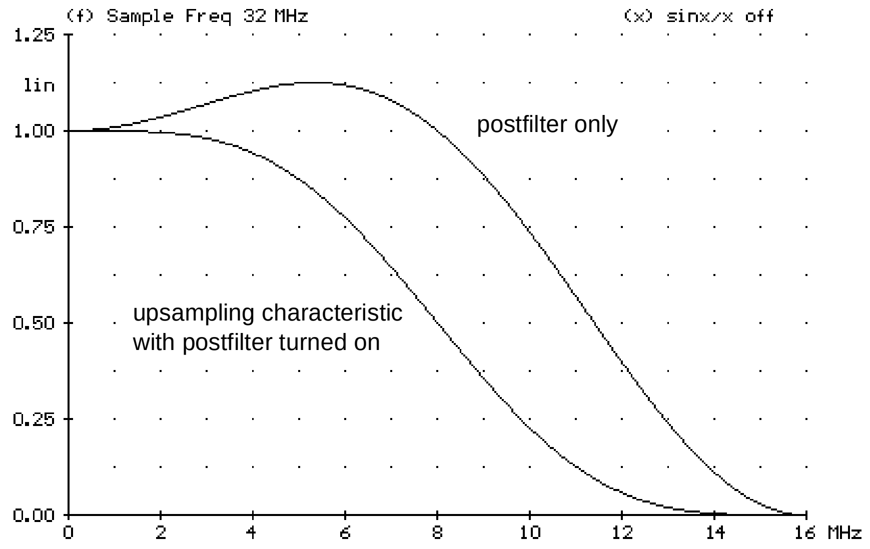


Fig. 33 Transfer curve of postfilter

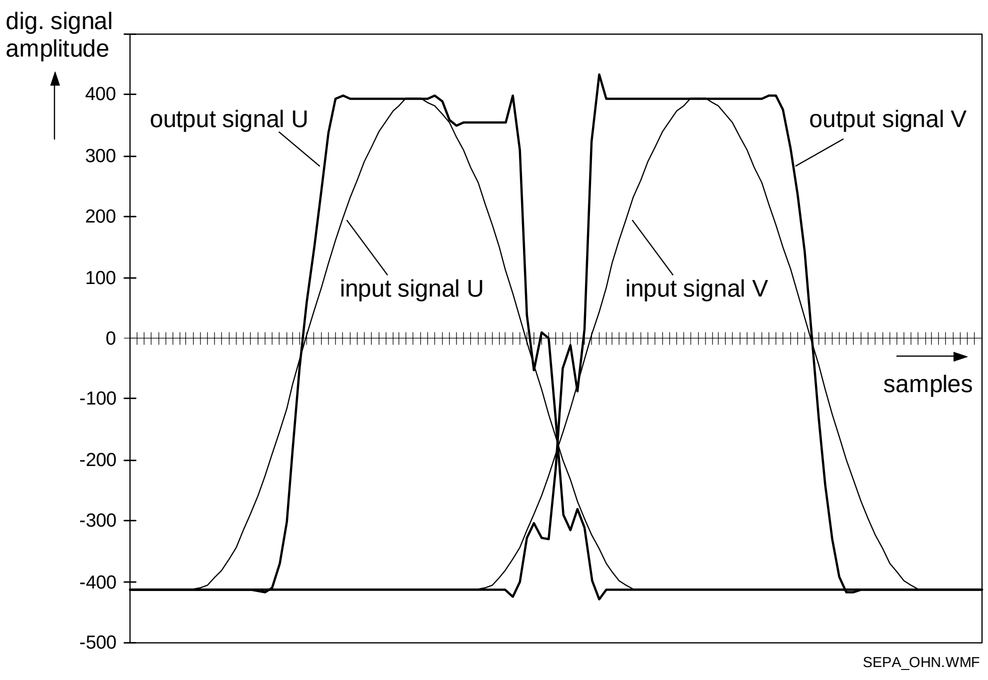


Fig. 34 DCTI with common processing of both signals (*CTI_SEPARATE* = 0)

The influence of each of the filters can be adjusted in eight steps from 0 to 8/16 (7/16 omitted). In fig. 37 to 39 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. 40 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

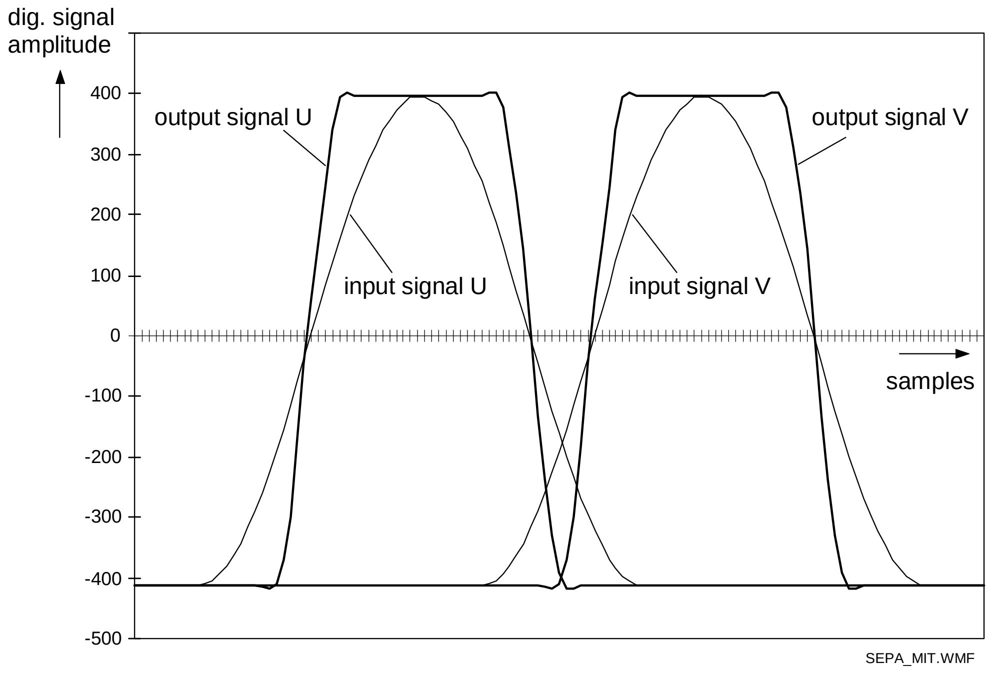


Fig. 35 DCTI with separate processing of both signals (separate = 1)

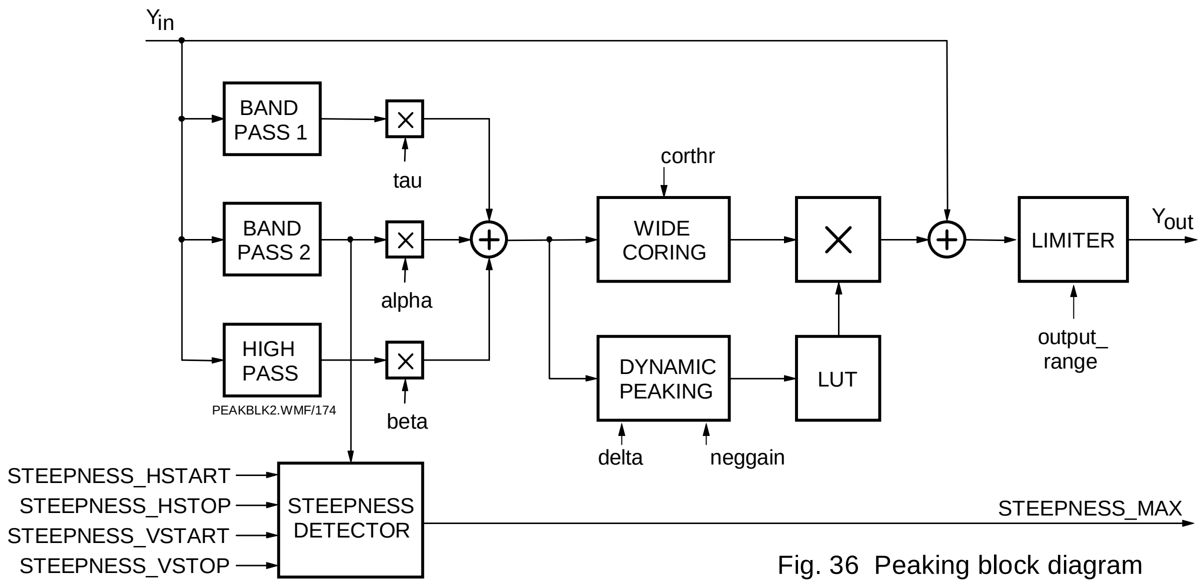


Fig. 36 Peaking block diagram

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. 41.

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 attenuator is shown in fig. 42. The value of 128 is equal to no attenuation. Maximum attenuation is achieved at value 24.

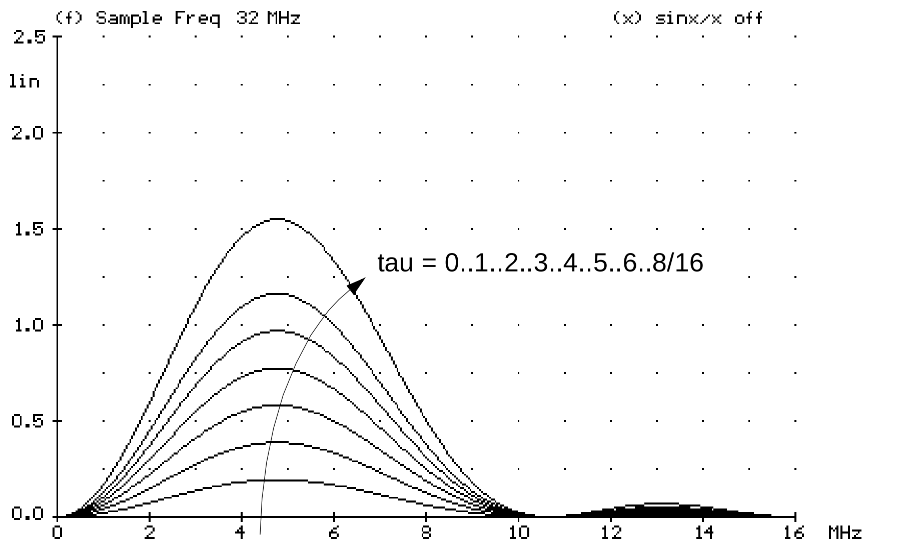


Fig. 37 Frequency response of the peaking band pass filter 1

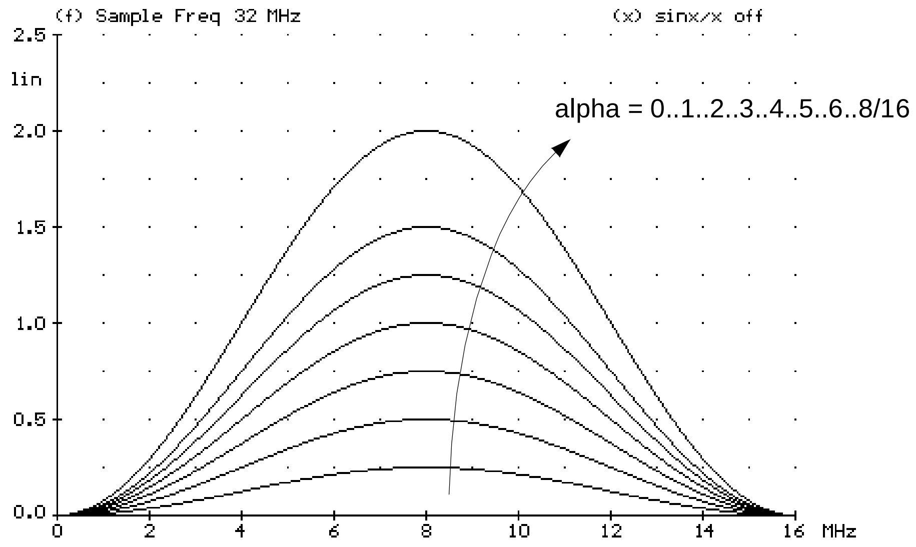


Fig. 38 Frequency response of the peaking band pass filter 2

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. 43 depicts the situation.

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

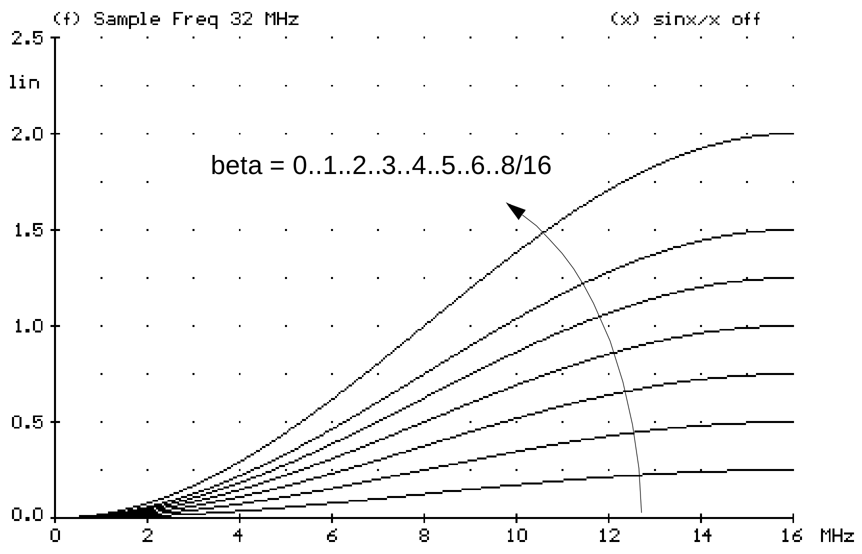


Fig. 39 Frequency response of the peaking high pass filter

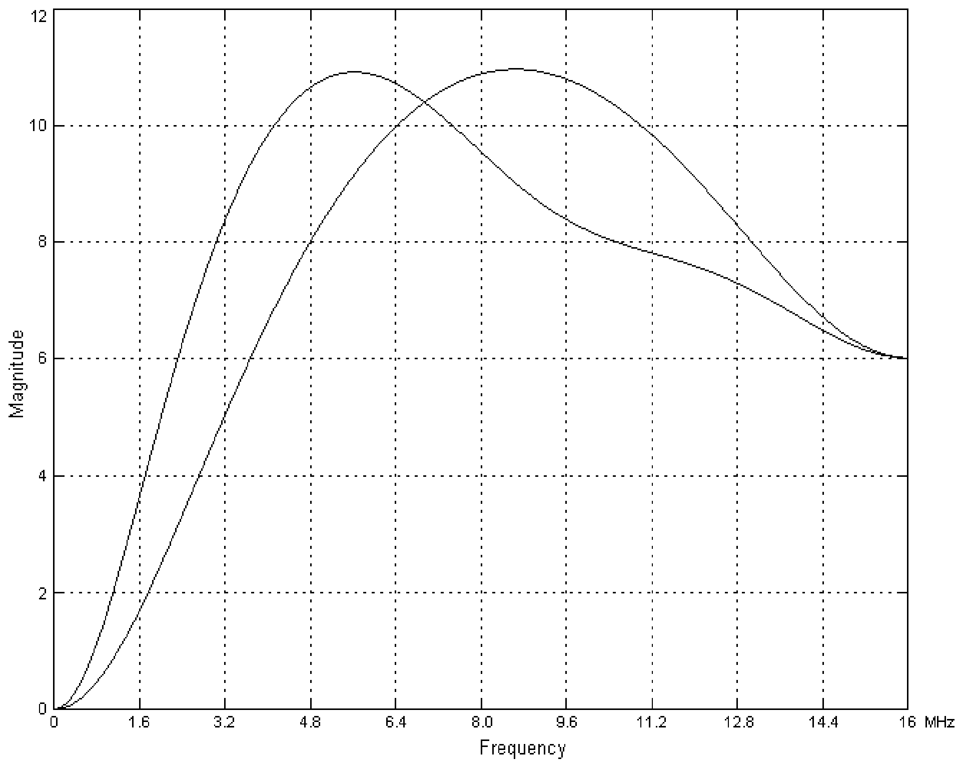


Fig. 40 Variation of peaking center frequency

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*.

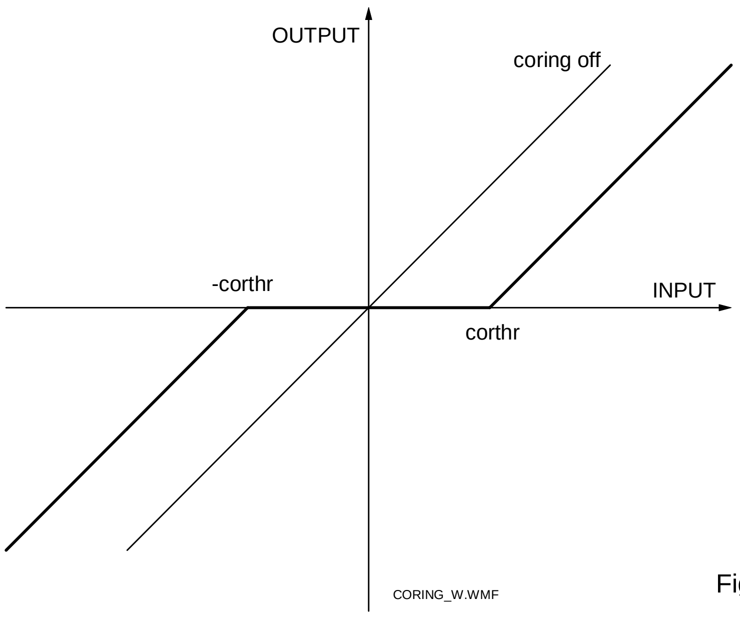


Fig. 41 Luminance coring

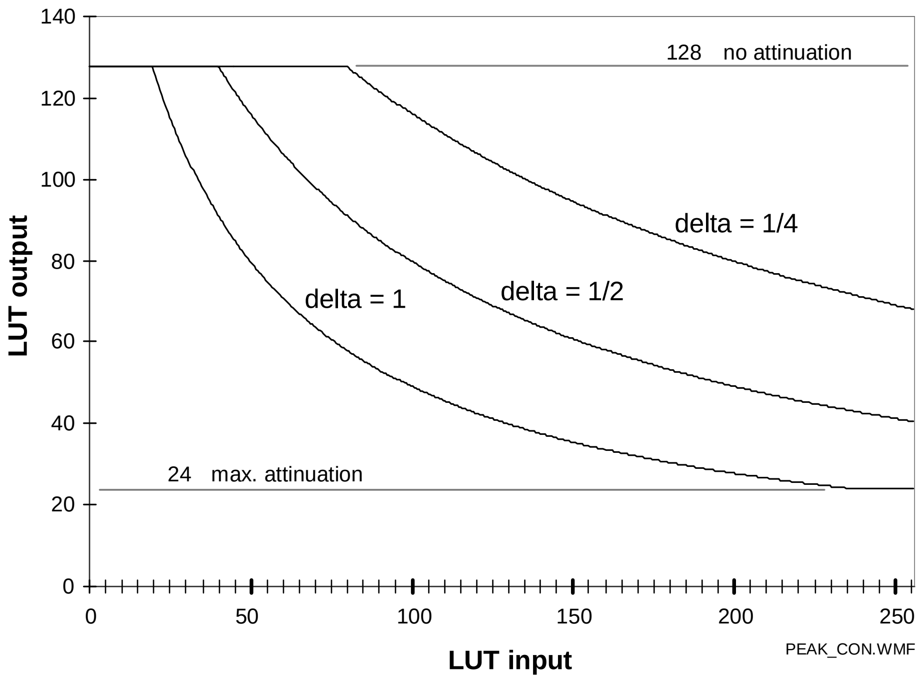


Fig. 42 Dynamic peaking control

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 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.

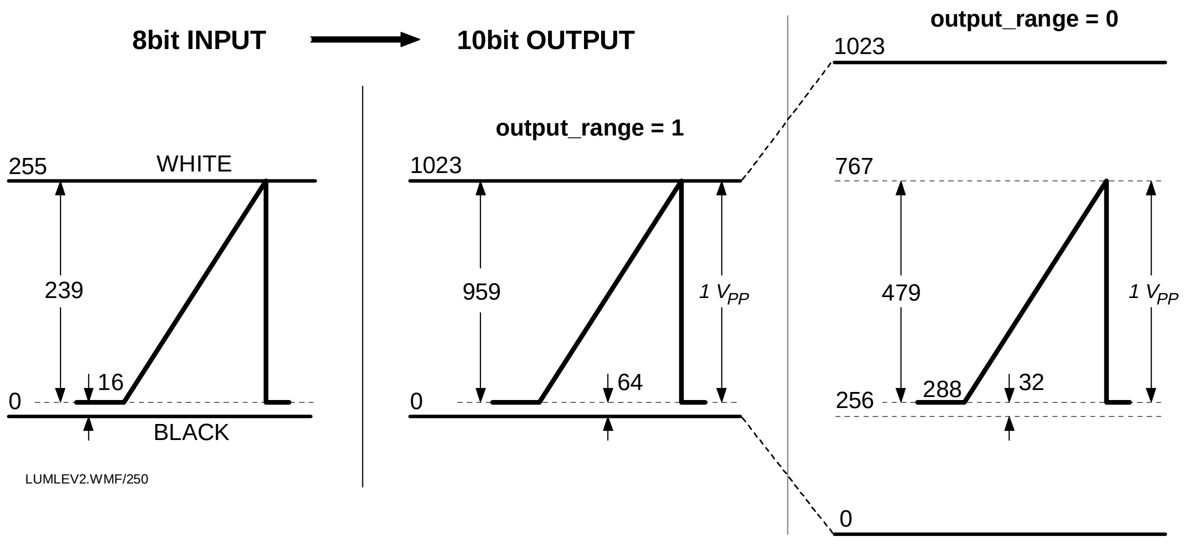


Fig. 43 Input / output signal levels of luminance signal

In fig. 44 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 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:

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

Fig. 45 NLP D/A gain settings

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

- side panels for 4:3 displays on a 16:9 screen
- 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. 46 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. 47 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

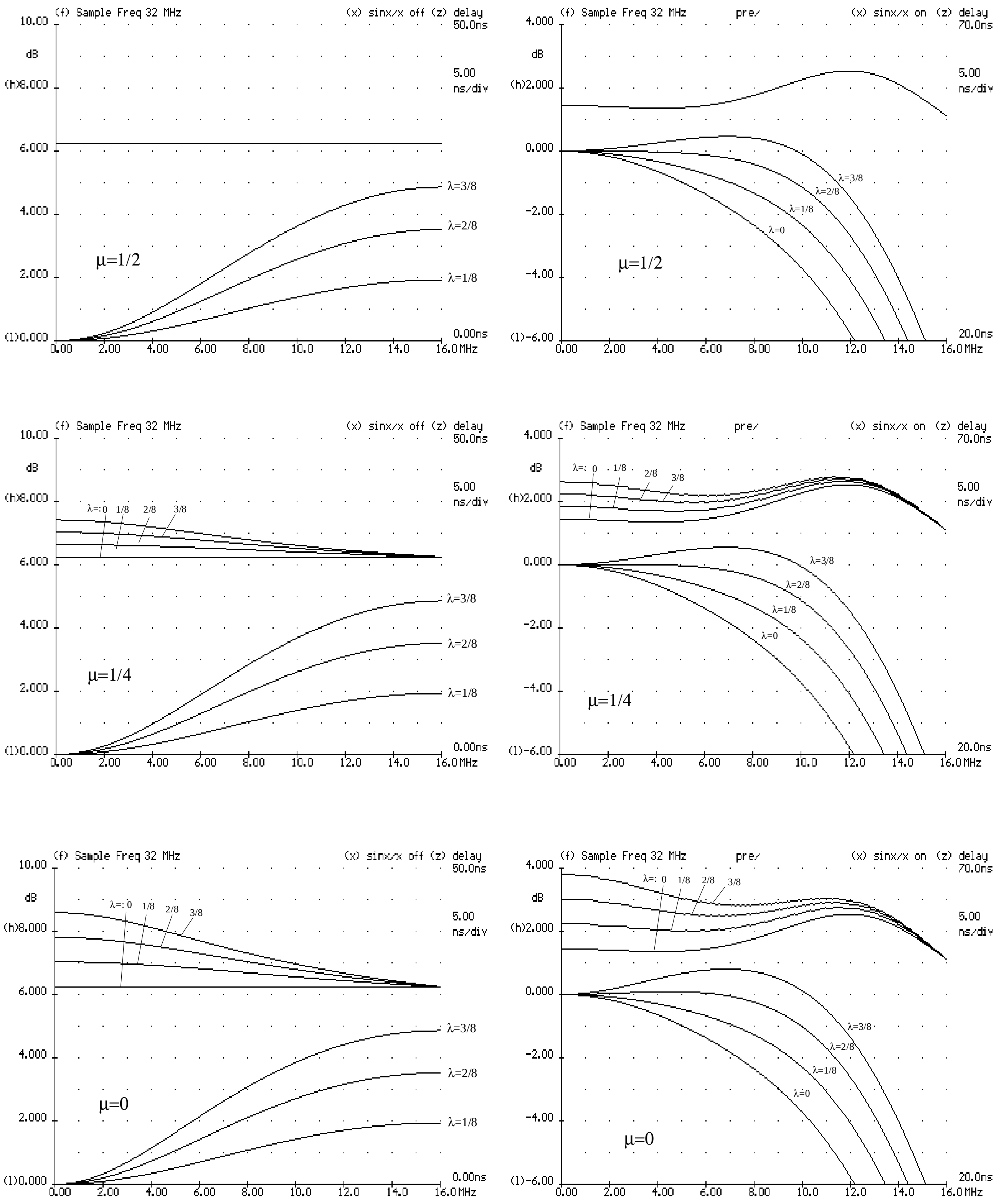


Fig. 44 Group delay and transfer curves of the NLP D/A filter

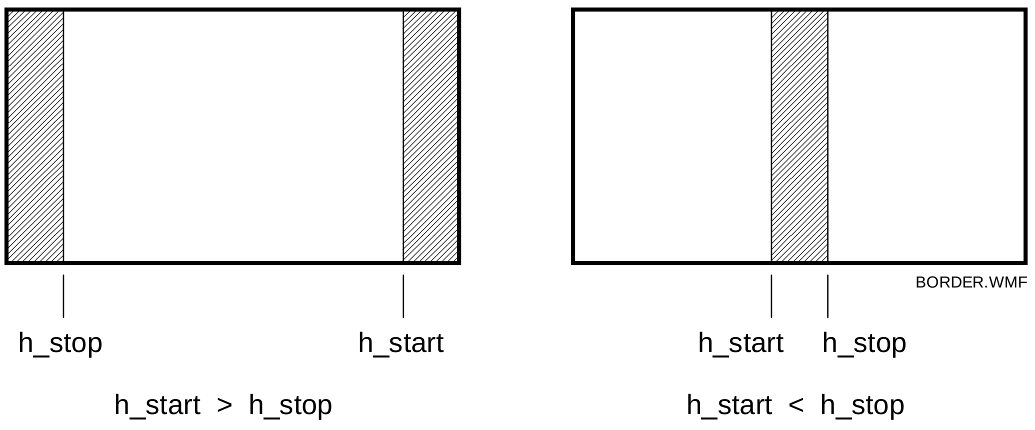


Fig. 46 Border definition

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. 47.

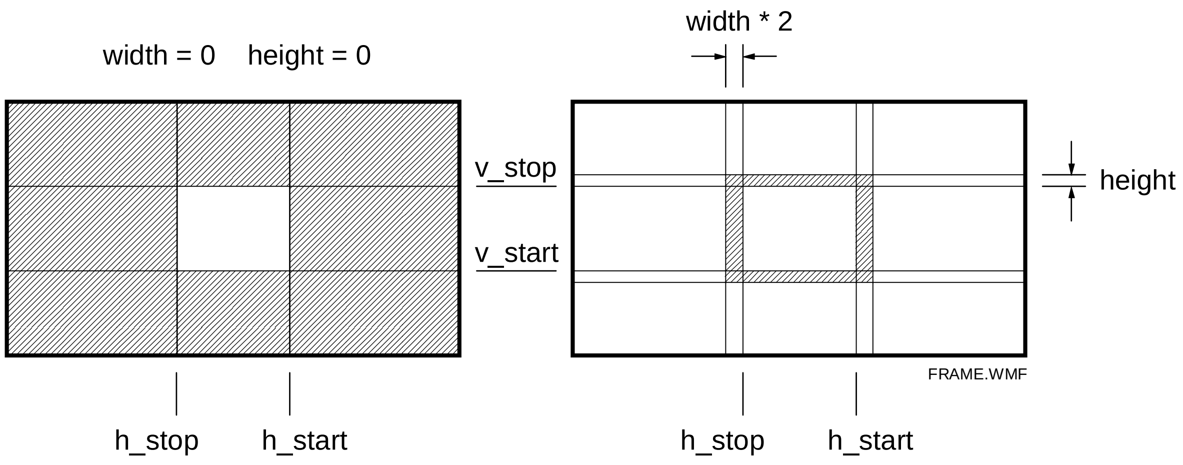
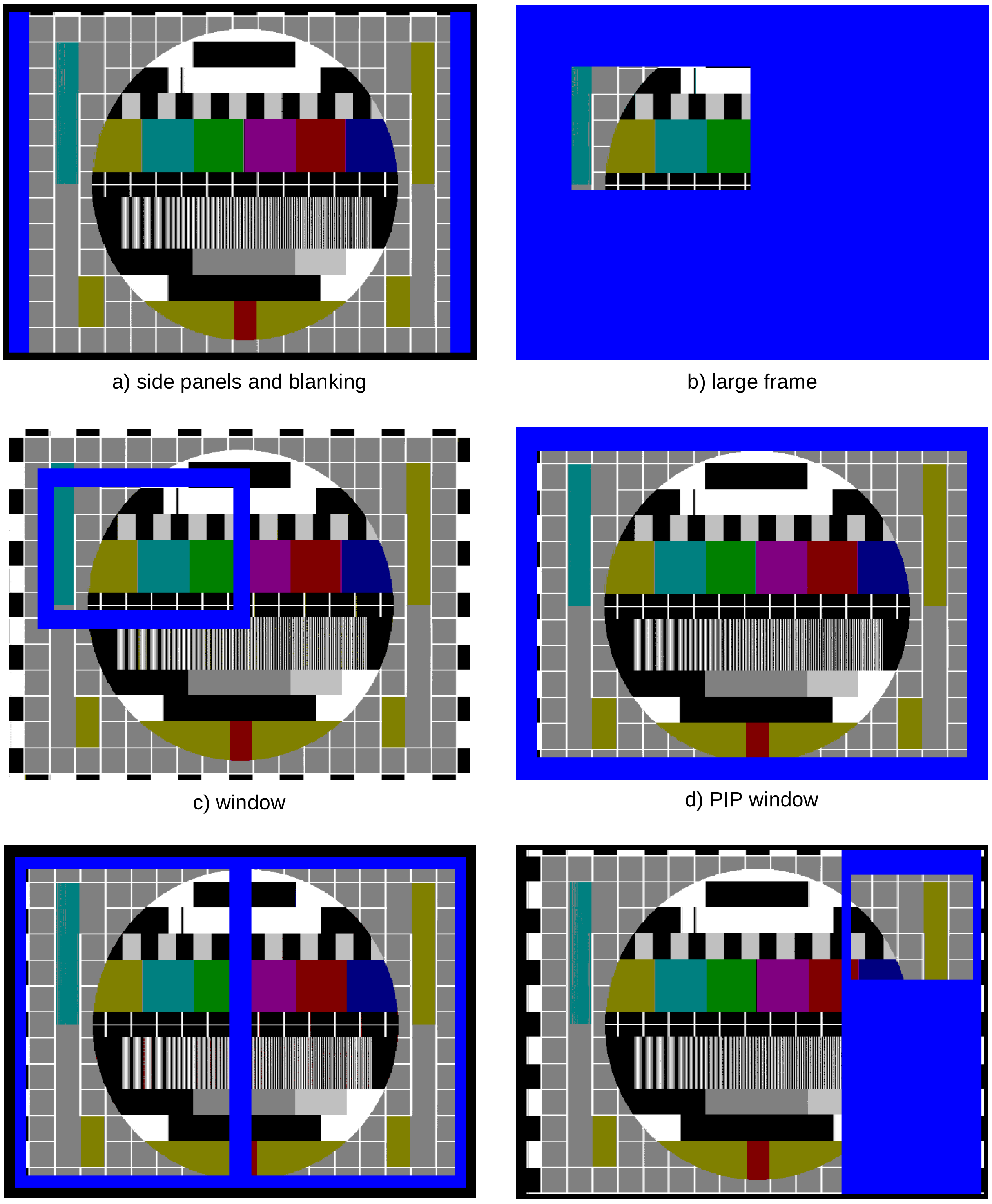


Fig. 47 Frame definition

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. 48:

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.

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



a) side panels and blanking

b) large frame

c) window

d) PIP window

e) POP and blanking

f) double window and blanking

Fig. 48 Examples for windows and frames

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. 49.

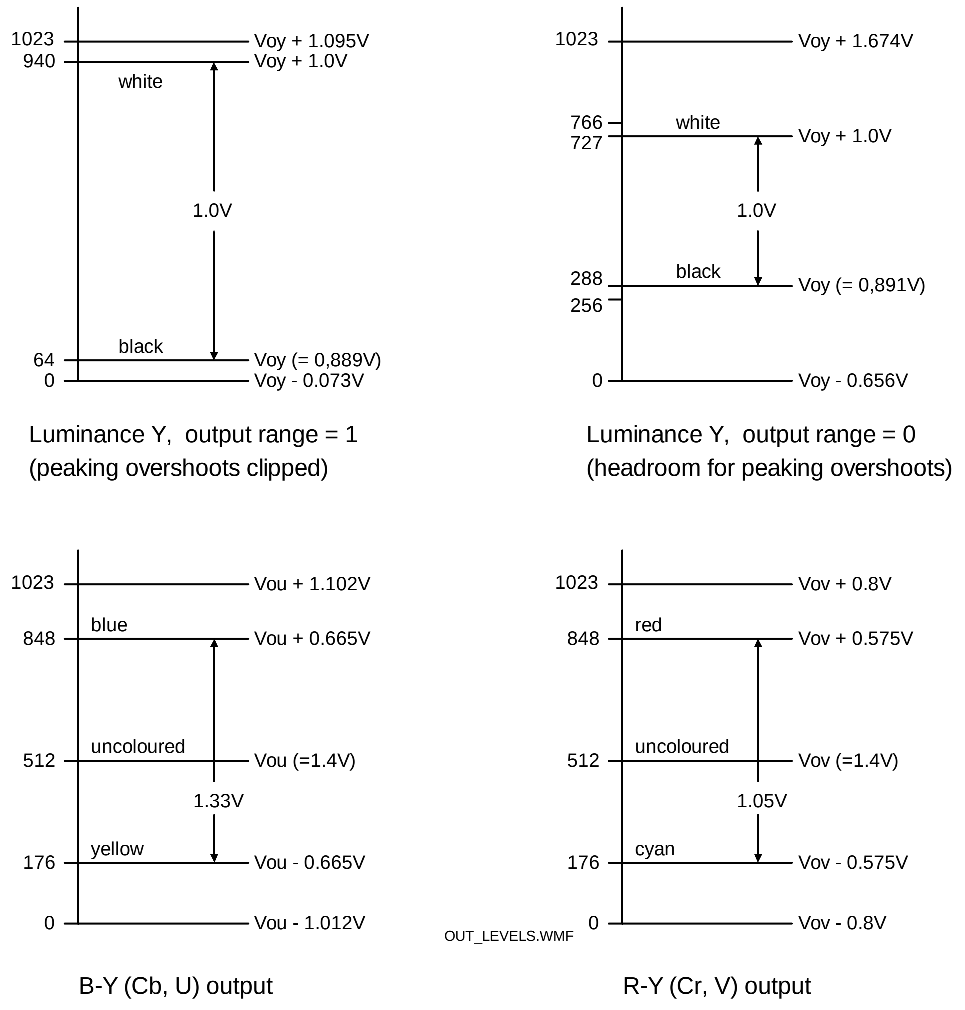


Fig. 49 Luminance and chrominance output levels

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 5.410, see chpt. 8.

For communication with external ICs two serial busses can be used, the I^{2}C bus and the SNERT bus. The I^{2}C-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]

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. 50.

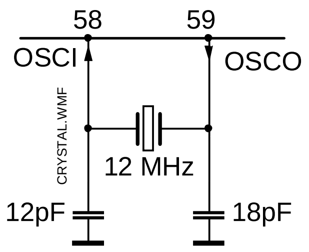


Fig. 50 Application diagram of Crystal for PLL

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

5. Functional description of the SAA4992

The key feature of the MK12 module - motion compensated field and line rate conversion - is realized by the IC SAA4992. It supports conversion to 100/120 Hz 2:1 (interlace) or 50/60 Hz 1:1 (progressive). Although not realized on the module, the IC 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).

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. 51. 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.

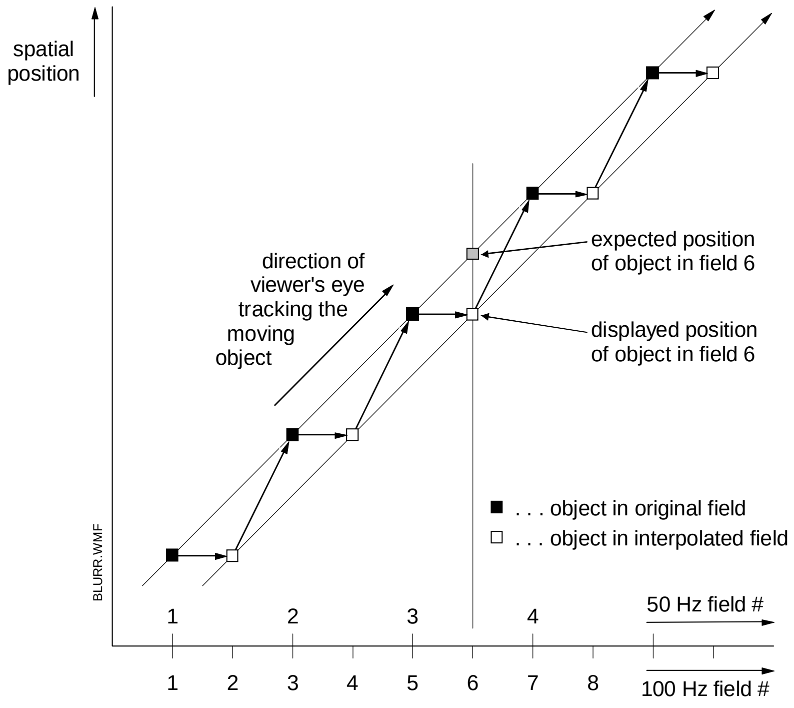

spatial position
direction of viewer's eye tracking the moving object
expected position of object in field 6
displayed position of object in field 6
■ . . . object in original field
□ . . . object in interpolated field
BLURR.WMF
1
2
3
4
50 Hz field #
1
2
3
4
5
6
7
8
100 Hz field #

Fig. 51 100 Hz field repetition causes blurring at moving edges

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.

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 SAA4992 is based on a 3-D recursive search block-matching algorithm. Fig. 52 demonstrates the 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

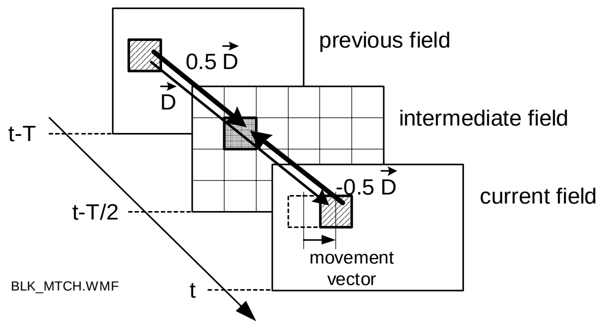


Fig. 52 Block matching principle

vertically ± 12 lines, horizontally ± 31.75 pixels (sub-pixel accuracy). The general architecture of the motion estimator and compensator in luminance is shown in the block diagram below (fig. 53).

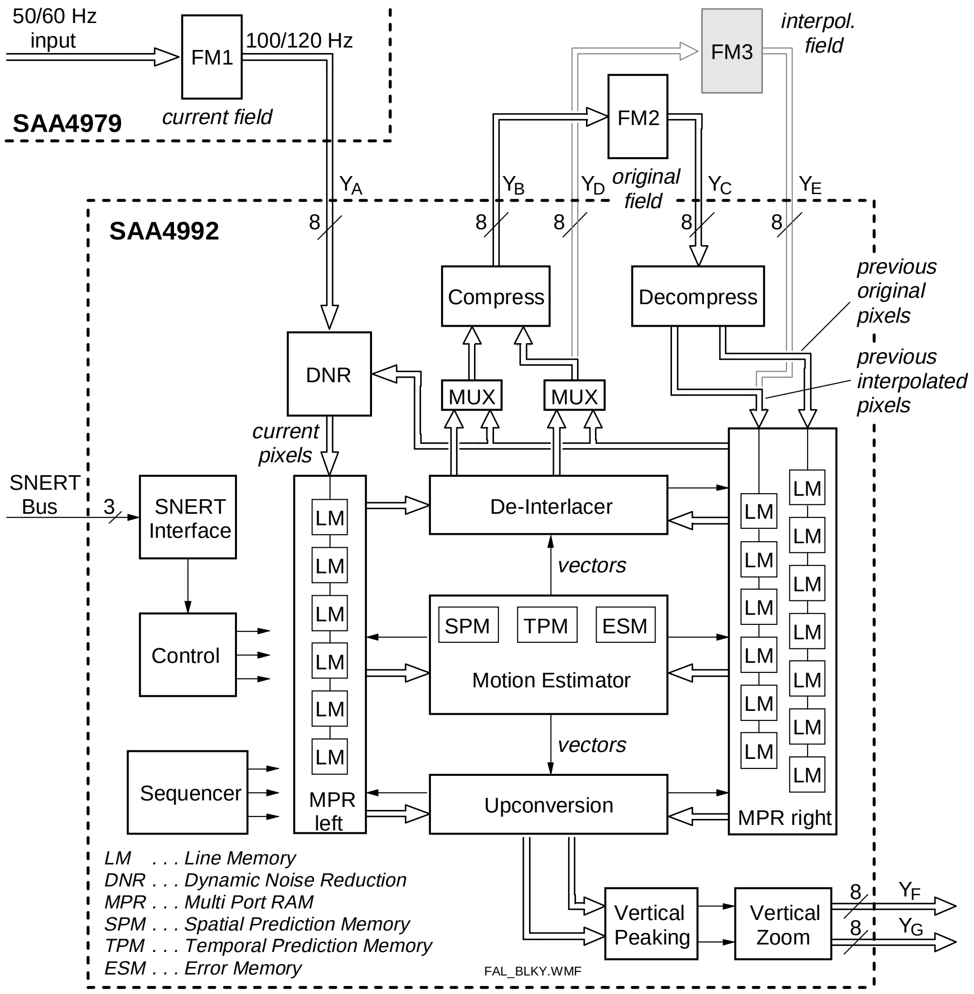


Fig. 53 Block diagram of the SAA4992 luminance processing

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 several line memories to hold the actual data, which are needed for 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, 7 sequential 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. 53.

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, 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 SAA4992 can be run without FM3. 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.

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. 54 show the principle.

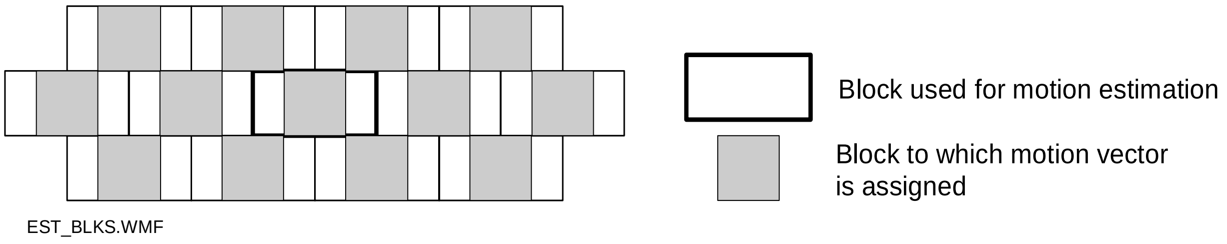

Block used for motion estimation
Block to which motion vector is assigned
EST_BLKS.WMF

Fig. 54 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. 55.

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

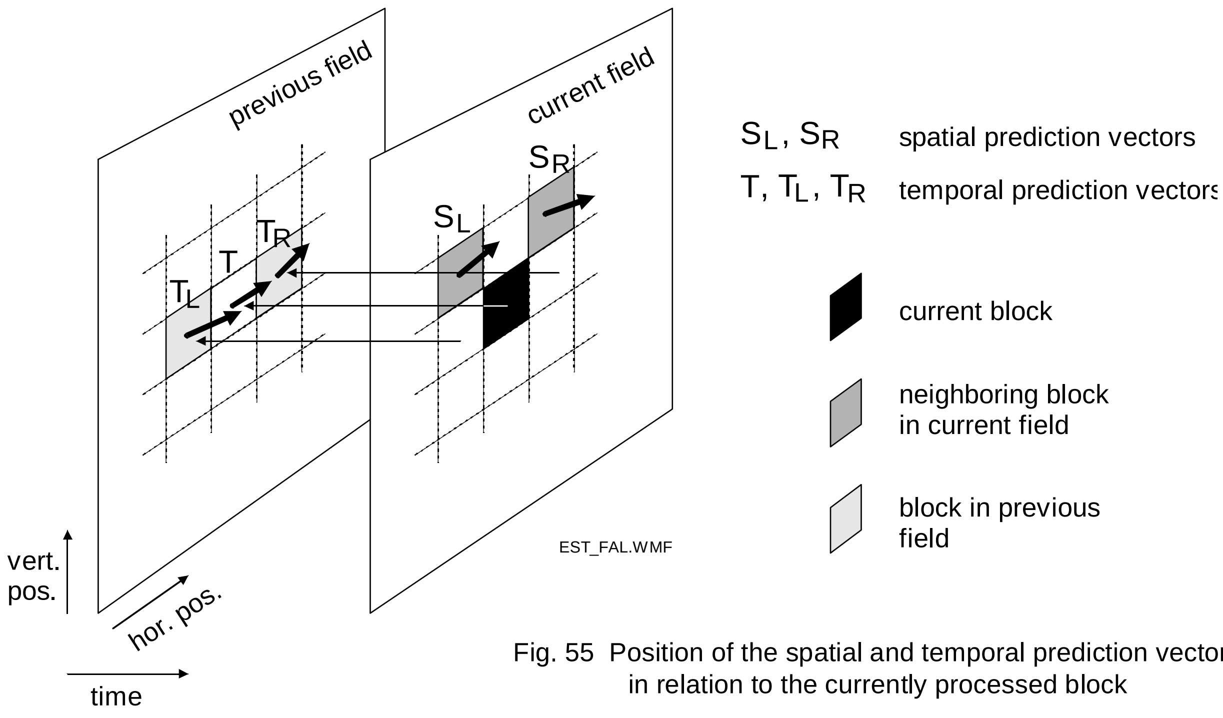


Fig. 55 Position of the spatial and temporal prediction vectors in relation to the currently processed block

- 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 1. U represents a random update vector which can be applied to any prediction vector.

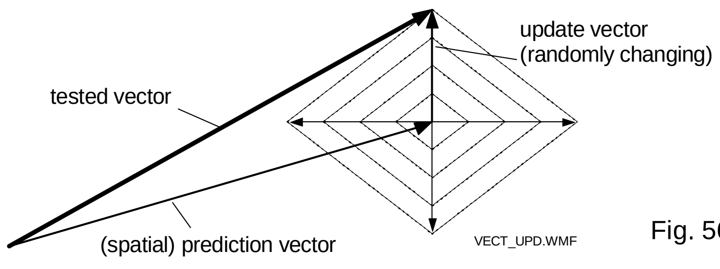

update vector (randomly changing)
tested vector
(spatial) prediction vector
VECT_UPD.WMF

Fig. 56 Recursive search trying to find a better vector

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. 57. 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.

C_P is a programmable vector. This candidate is possibly selected when camera panning or zooming is detected.

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 the right (R), which is marked in the column estimator in table 1. In hard-

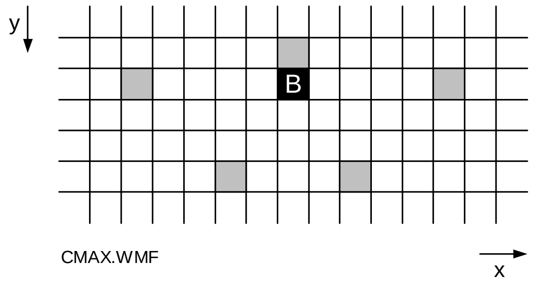


Fig. 57 Selection of C_{max}

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 1 Selection of vector candidates

ware only one motion estimator is used, which is multiplexed in time. Fig. 58 shows how in case of field rate doubling two estimations can be done for every input field.

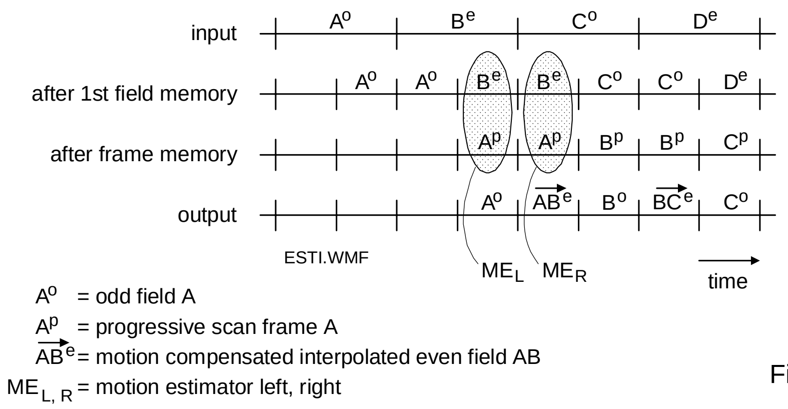

input
after 1st field memory
after frame memory
output
ESTI.WMF
ME_L
ME_R
time
A^o = odd field A
A^p = progressive scan frame A
$\overrightarrow{AB}^e$ = motion compensated interpolated even field AB
$ME_{L,R}$ = motion estimator left, right

Fig. 58 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. 59. 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).

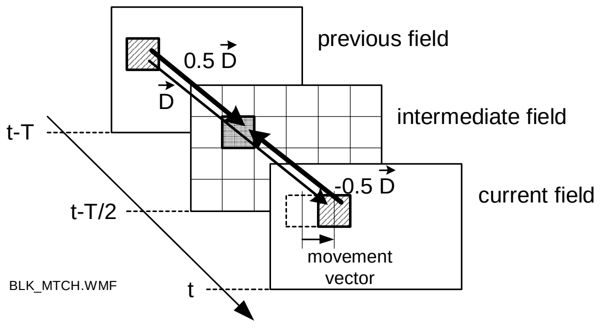


Fig. 59 Split vectors

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. 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)

In the temporal prediction memory (TPM) the vectors are stored which are calculated by the motion estimator, one for each block (4 lines, 8 pixels). Due to block subsampling (see fig. 54) only for every second block a motion vector is stored. A 4k x 16 bit SRAM, inside of the block, 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:

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

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 $D_{0\,0}$, $D_{0\,1}$, $D_{1\,0}$ and $D_{1\,1}$ 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. 60.

5.2.4 De-Interlacer

The task of the de-interlacer [also called PROgressive scan conversion (PRO)] is to convert the interlaced input signal to a progressive one. The de-interlaced picture data is stored in a frame memory (FM2 / FM3). In the de-interlacer 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.

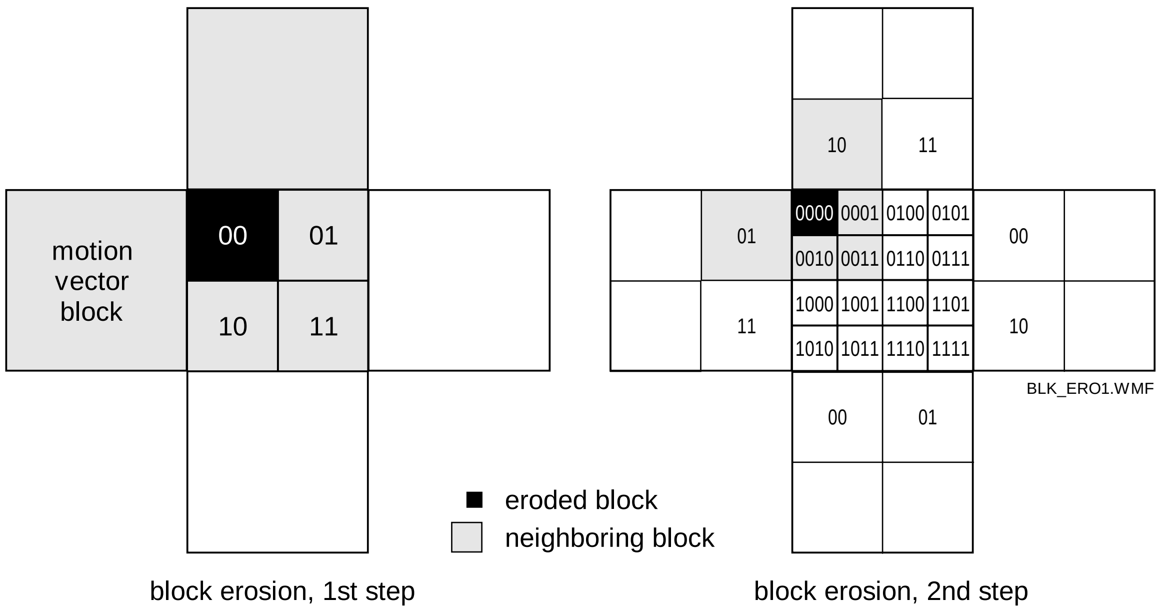


Fig. 60 Block erosion

The inputs for de-interlacer 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 in the left 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 de-interlacer is to add the interpolated lines to the original incoming lines (field in) and to mix them with motion compensated frame data.

The de-interlacer 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 the 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 pixel 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.

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. 61 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.

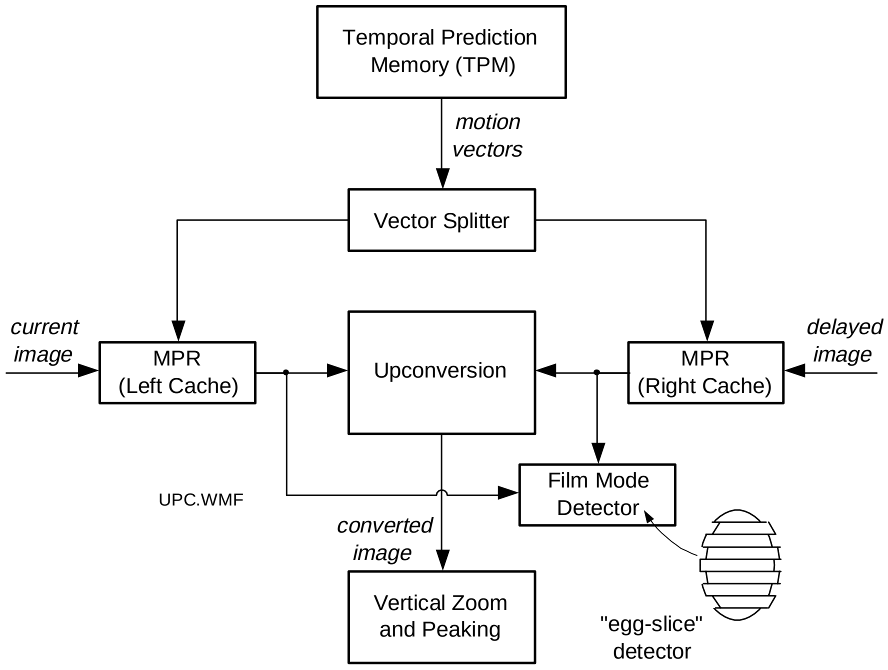


Fig. 61 Upconverter block diagram

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. 59 on page 50. 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.

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. 62 gives the transfer function of the peaking filter. Some of the 16 possible peaking coefficient steps are out of 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 matrix mode the peaking function should not be used as it will be undefined.

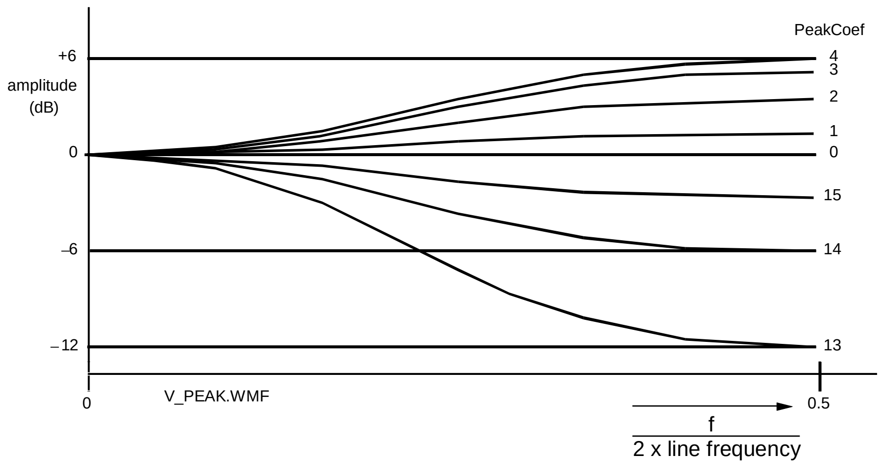


Fig. 62 Frequency response of the vertical peaking function

5.4 Luminance DPCM codec

Operating the SAA4992 requires two or three external field memories. In a two memory concept (without FM3) luminance can be compressed by a factor of 2 in a DPCM encoder and fed via bus Y_B into field memory FM2. The data from FM2 are processed by a DPCM decoder and supplied to the internal multi port RAM (right). Fig. 63 shows the data flow from the de-interlacer to FM2 and back to the right MPR.

5.5 Chrominance processing and memory configuration

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. 64.

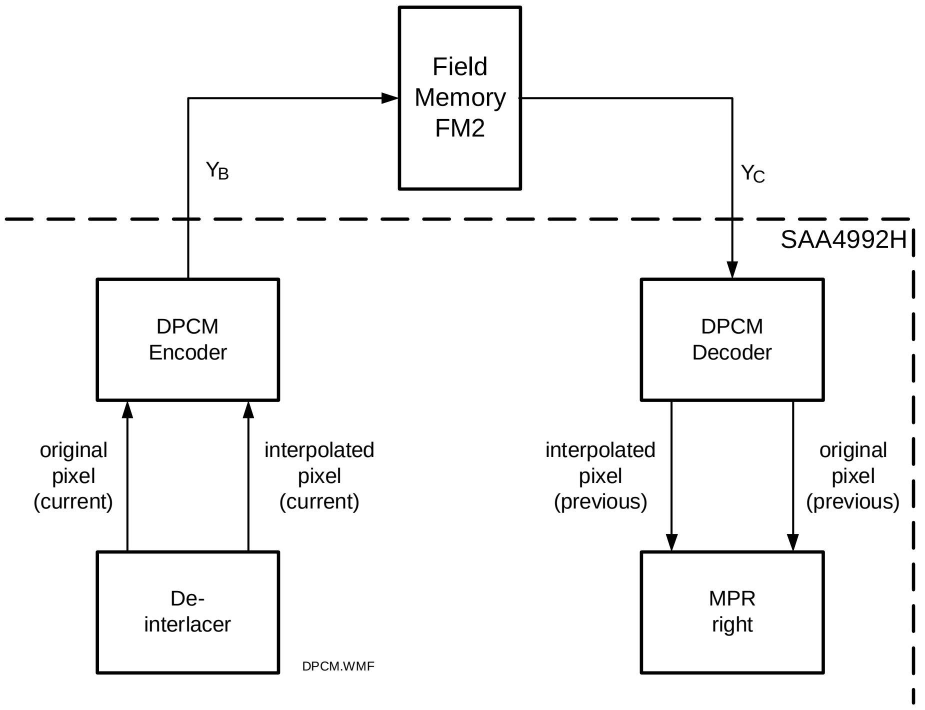


Fig. 63 Luminance DPCM data flow

FALCONIC can operate with two or three field memories. With three field memories the maximum performance is achievable. Then the configuration includes a field memory (FM1) at the input of Falconic plus a frame memory (FM2 + FM3) that contains frame data for luminance, but only field data in 4:2:2 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.

With 2 field memories, performance is reduced. The 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. Luminance is thereby compressed with 2-dimensional DPCM by a factor of 2, whereas color of 4:2:2 can be compressed with a factor 2 by (1-dimensional) DPCM. So in this 4:2:2 DPCM mode again the luminance of one frame and the chrominance of one field can be stored. If FM1 is only 12 bits wide, the only color options for the input are 4:1:1 or DPCM compressed 4:2:2, and formatting to true 4:2:2 takes place. This is the internally used data format.

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. Bus G is the optional matrix output to be used for $4\,f_H$ output. It is also 16 bits wide and can output data in the 4:2:2 format only.

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 can be used for evaluation purposes to have a real-time display together with the converted luminance of the vectors that have been estimated, but is also well suited for demonstration purposes.

5.6 Dynamic Noise Reduction (DNR)

As the block diagrams in fig. 53 and fig. 64 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.

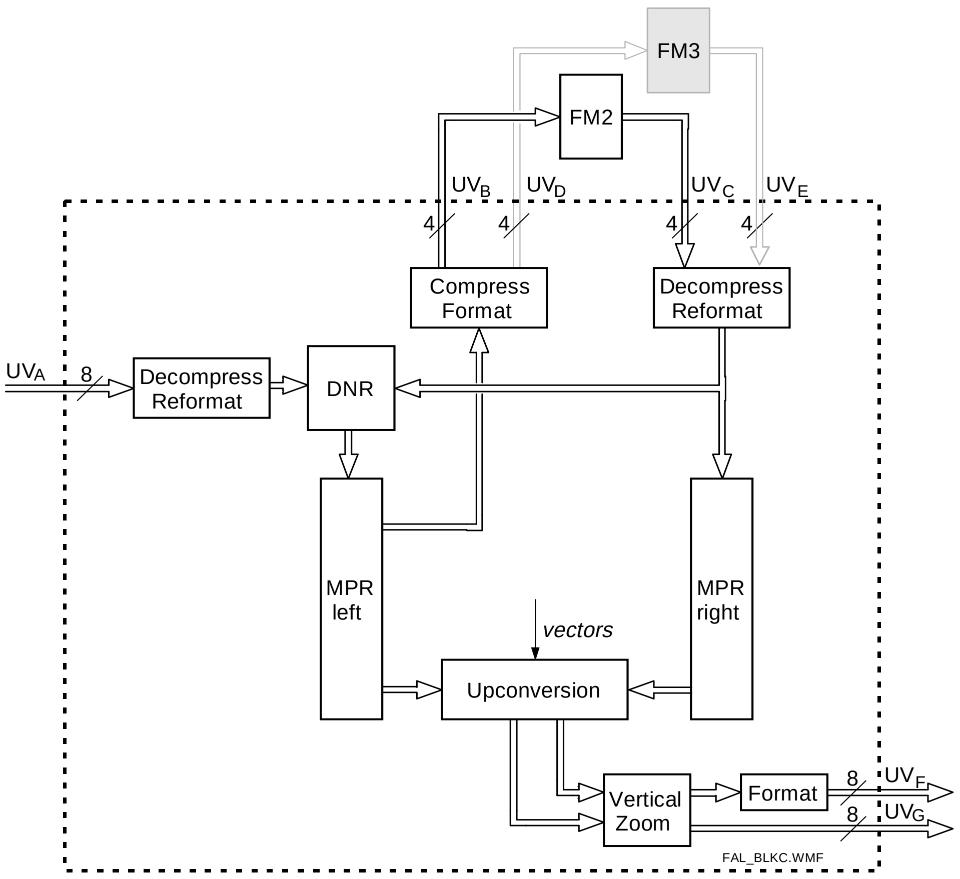


Fig. 64 Block diagram of chrominance processing

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. 65 gives a block diagram of the DNR circuit.

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.

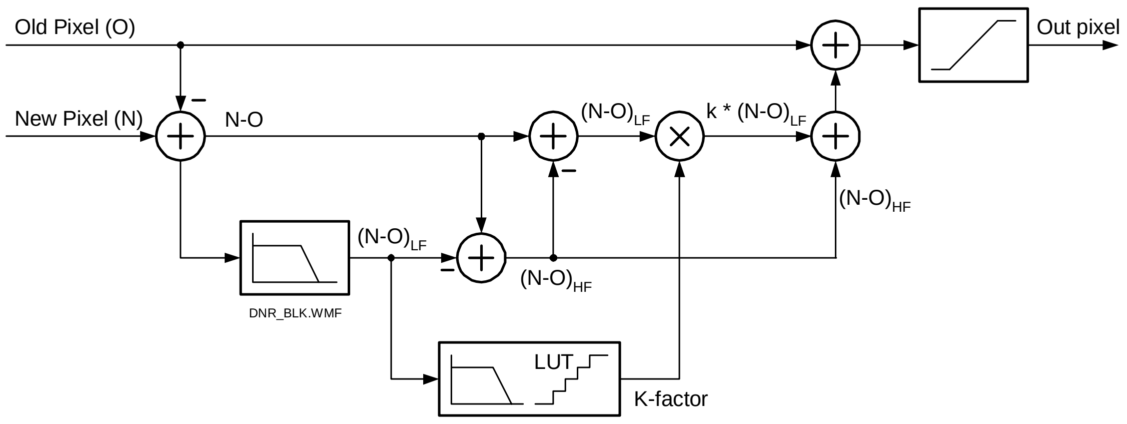


Fig. 65 DNR block diagram

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.

5.7 Comparison of DNR in the SAA4979 and SAA4992

On the MK12 module two noise reduction functions are available. They are basically identical. The few differences are listed in the table below:

TABLE 2 DNR comparison SAA4979 vs. SAA4992

	SAA4979	**SAA4992**
left over image artefact	eliminated due to ’noise shaping’	faintly visible
recursivity	field based	frame based
dirty window artefact		reduced by vector compensation

The main difference - and the main advantage of the DNR as implemented in the SAA4979 - is the noise shaping function. It effectively eliminates the problem that old images stay faintly visible on the screen at sudden scene changes.

It is therefore recommended to use the SAA4979’s noise reduction in all 100 Hz modes. In progressive scan mode however the noise reduction of the SAA4992 has advantages due to the frame based recursion.

6. Application environment

6.1 Motion compensation in a TV set

The IPQ module MK12 is intended to be used for scan conversion purposes in the TV environment. Fig. 66 gives an overview of a possible front end.

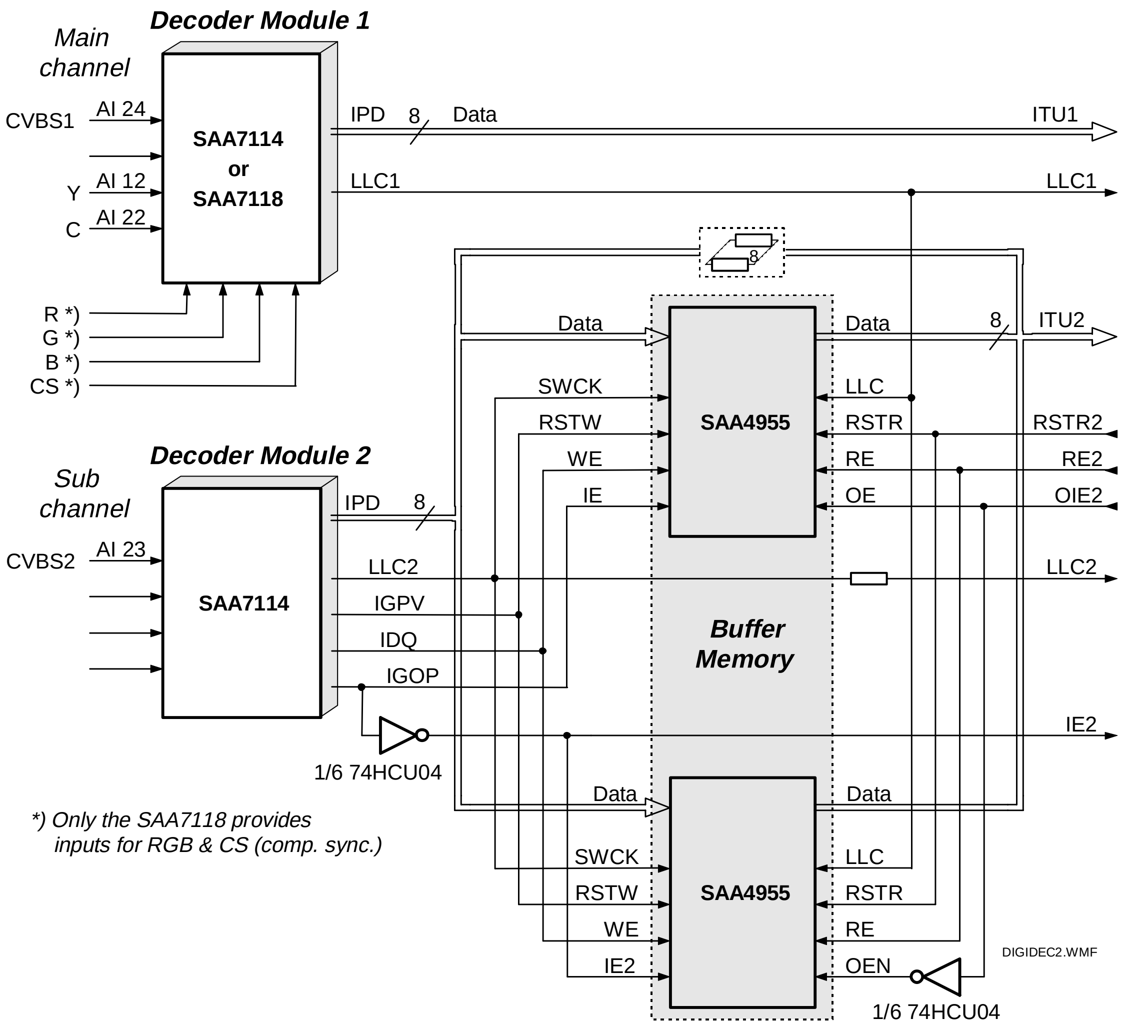


Fig. 66 Input environment for the IPQ module MK12

For decoding CVBS and S-Video signals as well as for input switching a DMSD (digital multi-standard decoder) is used for each input channel, either the SAA7118 or SAA7114. These decoders accept analog input signals and output a digital video data stream according to the ITU-656 standard. ITU1 is the main input 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 which serves as reference for the IPQ module’s main clock. The subchannel decoder is independent from the main channel one and writes its data into a buffer memory 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 on the MK12 module generates the memory control signals RSTR2, RE2 and OIE2. 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 of the two buffer memories data is read.

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, one memory could not 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.

6.2 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. 2-2 pull-down mode is depicted in fig. 67.

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 fig. 68. 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 the SAA4992. Smooth movement representatio without juddering is obtained. However the modes that the SAA4992 offers for this is either field rate doubling

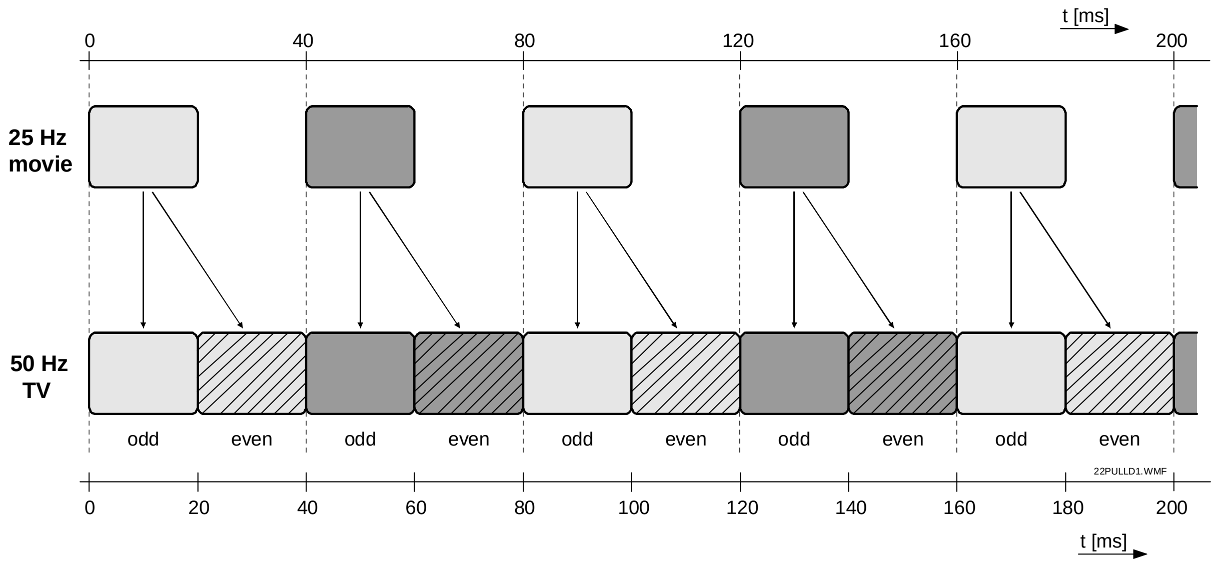


Fig. 67 Film to TV conversion in PAL (2-2 pull-down mode)

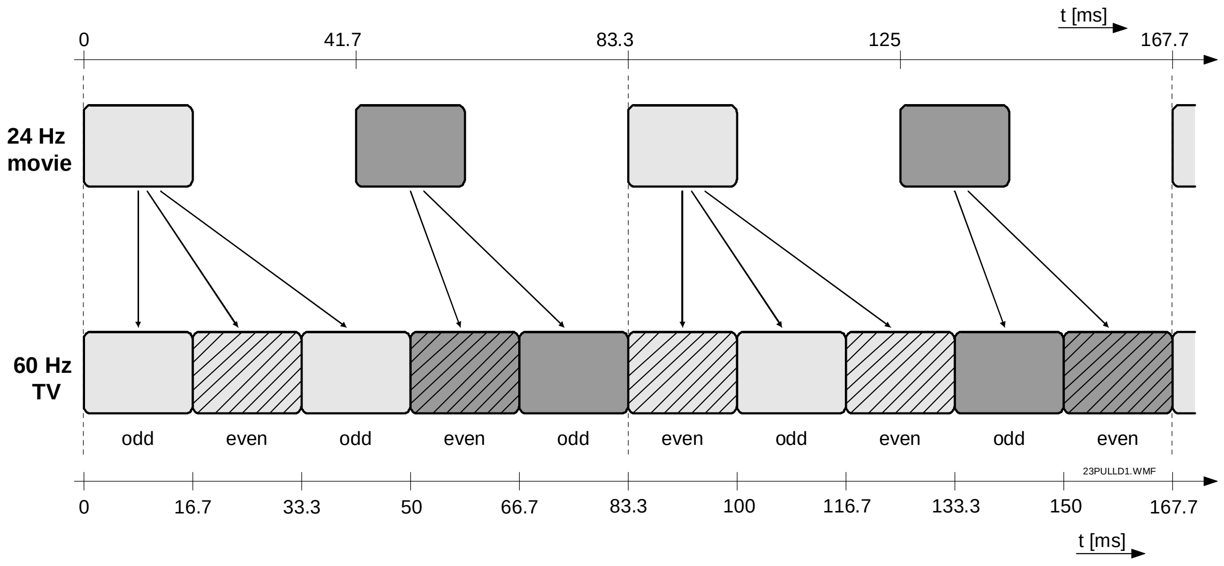


Fig. 68 Film to TV conversion in NTSC (3-2 pull-down mode)

(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 a deflection unit running on 32 kHz.

An attractive application would be to implement a movement compensation box directly in a DVD player. The output however would have to be in standard PAL or NTSC format, so any TV set (with 16 kHz deflection) could be used for display. Therefore an additional block is needed at the output of the SAA4992 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. 69.

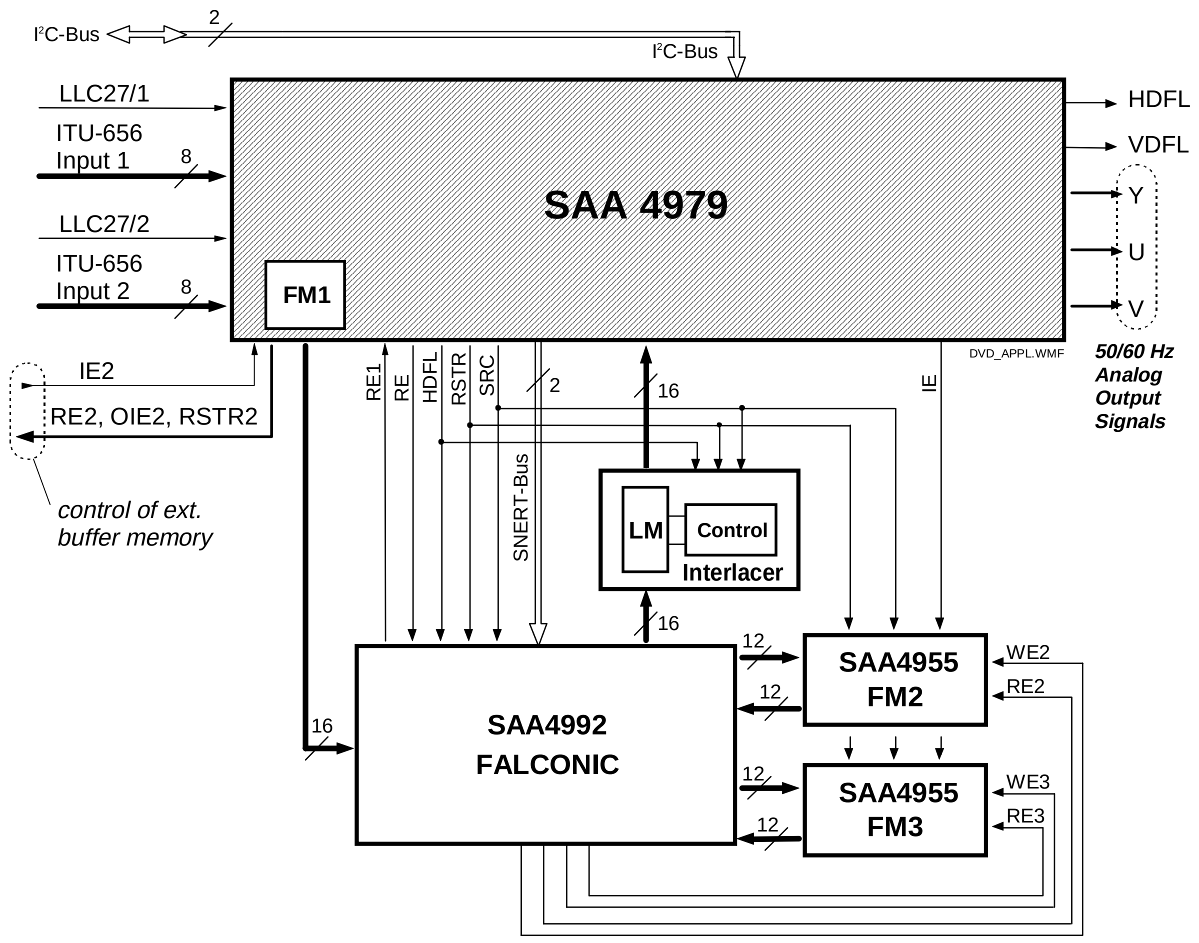


Fig. 69 Block diagram of motion compensation in a DVD player

7. PIP-Window construction using the PIP-Interface

7.1 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. 70.

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

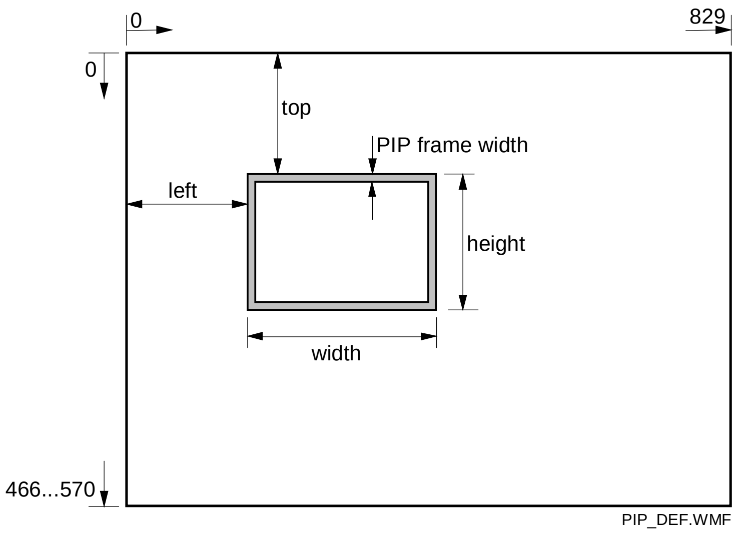


Fig. 70 Definition of the PIP-window

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.

7.2 PIP register definitions

7.2.1 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 sub-channel.
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.

7.2.2 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).

7.3 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).

7.4 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)

8. I^{2}C register tables

Firmware version: V 5.410
Date: March 14, 2001

The following tables describe the registers of the MK12 on-board microcontroller (here referred to as slave µC) which can be accessed via I^{2}C bus by the main controller (I^{2}C bus master controller).

When **writing** to the module the slave address is **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 **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 tranmitted (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 parantheses in the description column. This is the value that was loaded to the register at startup.

8.1 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	**Progressive Scan 1080 lines interlace; PSC, AFF need to be set** 0=off 1=on
	3	res_00_3	**reserved**
	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 1=on (one field out of AABB, full frame median filtered out of LFR)

	3..5	HZOOM_COMP_PAN	**Hzoom/Hcompression/Panorama** 0=off 1= 14:9 Horiz. Compress 2= 16:9 Horiz. Compress 3= Panorama 4= Amaronap 5= 14:9 Horiz. Zoom 6= 16:9 Horiz. Zoom 7= 22:9 Horiz. Zoom
	6..7	PP_COMP	**Picture Position HCompress** 0=centre 1=max left 2=max right
$02			**RESERVED_02 ($00)**
	0..7	res02_0_7	reserved for further modes

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: SCF via Sidepanels (default) 1: SCF via H-Blanking Enable SCF 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 by steps of 1/32
	6	VCOMPR	**Vertical compress bit** 0=default 1=vertical compression via VZOOM

	7	DIS_Feat	**Feature Mode** 0= normal mode 1= disable Feature Mode

NR 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	**state of SlaveUVtoY** 0=Default 1=SlaveUVtoY
	4..5	SCF	**Screen fade** 0=off 2=fade in 3=fade out
	6	DebugMode	Debug Mode (sidepanel color for: video = red; movie23 = green; movie22 = blue; phase 22 = blue/grey. Attention: please set bit SET_SIDEP ($0A) to enable the sidepanel and set the size of the sidepanels ($2D, $2E, $35).) 0=Debug Mode is disable (Default) 1=use of sidepanels to show the current processed mode
	7	DME	Detected Mode Evaluation; compare the phase of current and previous 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 (bit 0..7) two pixel delay to blank the TRC-bytes of the incomimg ITU datastream
$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			**Blank_fields ($20)**
	0	BLANK_F0	**field 0 (only activ if 'Set_Bln' is zero!)** 0=active 1=blanked
	1	BLANK_F1	**field 1 (only activ if 'Set_Bln' is zero!)** 0=active 1=blanked

	2	BLANK_F2	**field 2 (only activ if 'Set_Bln' is zero!)** 0=active 1=blanked
	3	BLANK_F3	**field 3 (only activ if 'Set_Bln' is zero!)** 0=active 1=blanked
	4	Embrace_off	**Embrace Mode** 0=Embrace Mode 1=No Embrace Mode
	5	V52_HControlOff	**V52_HControl** 0 = V5.2 H Control values used 1 = V5.3 H Control values used
	6	Set_Bln	enable hbln_sta, hbln_sto, vbln_sta, vbln_sto, hbln_vbln_MSB ($30, $31 $32, $33, $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-Bandwidthdetection** 0=default 1=UV Bandwidthdetection on
	7	res09_7	**reserved**

DIVERSE_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	SET_VADS	Enable direct access for VADSSTA,VADSSTO,HAD_VAD_MSB ($0E, $0F, $10) 0=off 1=on
	3	SET_HADS	Enable direct access for HADSSTA,HADSSTO,HAD_VAD_MSB ($0C, $0D, $0E) 0=off 1=on
	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 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	SMOOTH_LOCK	SMOOTH_LOCK
	1	CRN_DIRECT	CRN_DIRECT
	2	VRES_DIS	VRES_DIS
	3	res0B_3	reserved
	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 (enable via SET_HADS, sub 0Ah)
$0D			**HADSto ($00)**
	0..7	HADSSTO_0_7	Stop of hor. Auxiliary Displ. Signal LSB (enable via SET_HADS, sub 0Ah)

$0E			**HAD_VAD_MSB ($00)**
	0..1	HADSSTA_8_9	Start of hor. Auxiliary Displ. Signal MSB (enable via SET_HADS, sub 0Ah)
	2..3	HADSSTO_8_9	Stop of hor. Auxiliary Displ. Signal MSB (enable via SET_HADS, sub 0Ah)
	4..5	VADSSTA_8_9	Start of vert. Auxiliary Displ. Signal MSB (enable via SET_VADS, sub 0Ah)
	6..7	VADSSTO_8_9	Stop of vert. Auxiliary Displ. Signal MSB (enable via SET_VADS, sub 0Ah)
$0F			**VADSta ($00)**
	0..7	VADSSTA_0_7	Start of vert. Auxiliary Displ. Signal LSB (enable via SET_VADS, sub 0Ah)
$10			**VADSto ($00)**
	0..7	VADSSTO_0_7	Stop of vert. Auxiliary Displ. Signal LSB (enable via SET_VADS, sub 0Ah)

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 (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=off 1=on (enable via SET_NR, SEL_NR, sub 0Ah)
$16			**KCHROMA ($00)**
	0..3	kchromafix	kchromafix (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=off 1=on (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..7	res17_5_7	reserved

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 ($00)**
	0..3	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)
	4	res1A_4	reserved
	5	SlavUVtoY	**Slave UV to Y in reg. DnrColorMode** 0=off 1=on (enable via SET_NR, SEL_NR, sub 0Ah)
	6	DnrSplit	**Dnr split in reg. DnrColorMode** 0=off 1=on (enable via SET_NR, SEL_NR, sub 0Ah)
	7	DnrHpOn	**DnrHpOn in reg. DnrColorMode** 0=off 1=on (enable via SET_NR, SEL_NR, sub 0Ah)

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 activ; if value is $FF then mode is disabled; Mode must always 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 (without motion compensation, TRUE-movie mode); if this value is $00 then mode is always activ; if value is $FF then mode is disabled; Mode must always enabled by the bit 'MOVIE_FALLBACK' in subaddress $00! Attention! This threshold only works in movie23 (FALCONIC)

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..7	res1D_3_7	reserved

NOISE_EST_WRITE

$1E			**compns_ypscale ($01)**
	0..1	ypscale	Y scaling factor for prefilter
	2..5	compensate	compensate value can be added to NEST
	6..7	res1E_6_7	reserved
$1F			**gain_sob_clip_ne ($01)**
	0..2	gain_upbnd	set gain of upper boundary
	3	sop_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	res1F_7	reserved
$20			**wanted_value ($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..7	res27_2_7	reserved

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	res28_1	reserved
	2..5	HPeaking_Curve	**HPeaking_Curve** 0=off 1…F= curves 1…15
	6..7	res28_6_7	reserved

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** 0=off (Default) 1=on
	3	dcti_protect	**DCTI protec** 0=off 1=on (Default)
	4	dcti_postfilter	**DCTI postfilter** 0=off 1=on (Default)
	5	dcti_superhill	**DCTI superhill** 0=off 1=on (Default)
	6..7	res2A_6_7	reserved

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..7	res2F_6_7	reserved
$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..6	res37_0_6	reserved
	7	PLL_open	PLL_open 0=default (closed) 1=PLL_open (enable via SET_PLL, sub 0Ah)

PIP INTERFACE

$38			**PIP_Vtop ($00)**
	0..7	PIP_Vtop_0_7	Vertical postion of the high left corner of PIP LSB
$39			**PIP_Hleft ($00)**
	0..7	PIP_Hleft_0_7	Horizontal position of the high left corner of PIP LSB
$3A			**PIP_H_V_MSB ($00)**
	0..1	PIP_Vtop_8_9	Vertical postion of the high left corner of PIP
	2..3	PIP_Hleft_8_9	Horizontal position of the high left corner of PIP
	4..5	PIP_Hwidth_8_9	Width of PIP window
	6..7	PIP_Vhigh_8_9	High 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	High 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 activ)
	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..REG3C to PIP frame **Attention!** If this value is not equal zero the whole PIP interface fucntionality is disabled
	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 could be dynamically changed (NOT for Multi-PIP!)
	6..7	res3F_6_7	reserved
$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 ($8C)**
	0..3	avgMovieProcessing	average value of the movie processing time; if the current movie mode processing time is less, the next switch to movie mode would be delayed. This value should be set to 1.2s (resolution is 1/10s)
	4..7	lastVideoTime	if the time of the last processed movie mode is less than this value, the switch to movie mode would be delayed. This value should be set to 0.8s (resolution is 1/10s)
$42			**movieSwitchSettings_2 ($98)**
	0..3	oftenSwitchDelayPAL	if a switch to movie happens too often, the switch is delayed. This value should be set to 8.0s (resolution is 1s)
	4..7	oftenSwitchDelayNTSC	if a switch to movie happens too often, the switch is delayed. This value should be set to 0.9s (resolution is 1/10s)

$43			movieSwitchSettings_3 ($55)
	0..3	wrongPhaseDelay	if the second detected movie mode has not the same phase as the previously detected one, the mode switch would be delayed. This value should be set to 5.0s (resolution is 1s)
	4..7	fastSwitchDelay	The first detected movie mode is never delayed but a delay timer is started. If the delay timer is running, the phase and the average movie mode processing time is evaluated before a next switch to movie mode can occur. This value should be set to 20s (resolution is 4s).

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)

Write Factory Settings

$99			WriteFactorySettings ($00)
	0	Show_Reset_Write	only use as read value in master software
	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 of hardware concept 0 = use automatic detected value (default) 1 = force BESIC422 only 2 = force BESIC422 / RAVEN 3 = force BESIC422 / FALCONIC
	5..7	res99_5_7	reserved

8.2 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	Nm_Active	Naturel motion 0=normal mode 1=Natural 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..7	resrd01_6_7	reserved

SOFTWARE_VERSION

$02			**SOFTWARE_VERSION ($54)**
	0..7	SOFTWARE_VERSION	version of the slave control software

Hardware_ID

$03			**Hardware_ID ($0F)**

	0..7	Hardware_ID	Hardware Id. (50ms < concept detection < 650ms) The following concepts are supported:
			0Bh = only SAA4979
			EBh = SAA4979 / SAA4992 (Raven)
			FBh = SAA4979 / SAA4992

NoiseEstimation

$04			**noise_estimation ($00)**
	0..3	nest	noise estimation
	4..7	resrd04_4_7	reserved
$05			**noise_estimation_filter ($00)**
	0..7	nest_filt	noise estimation filter
$06			**detail counter MSBs ($00)**
$07			**detail counter LSBs ($00)**
	0..7	detail_cnt	detail counter
$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	resrd09_2_7	reserved

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	Quit_Reset_Read	only in write mode
	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	resrd99_5_7	reserved

9. Circuit diagrams of the IPQ module MK12

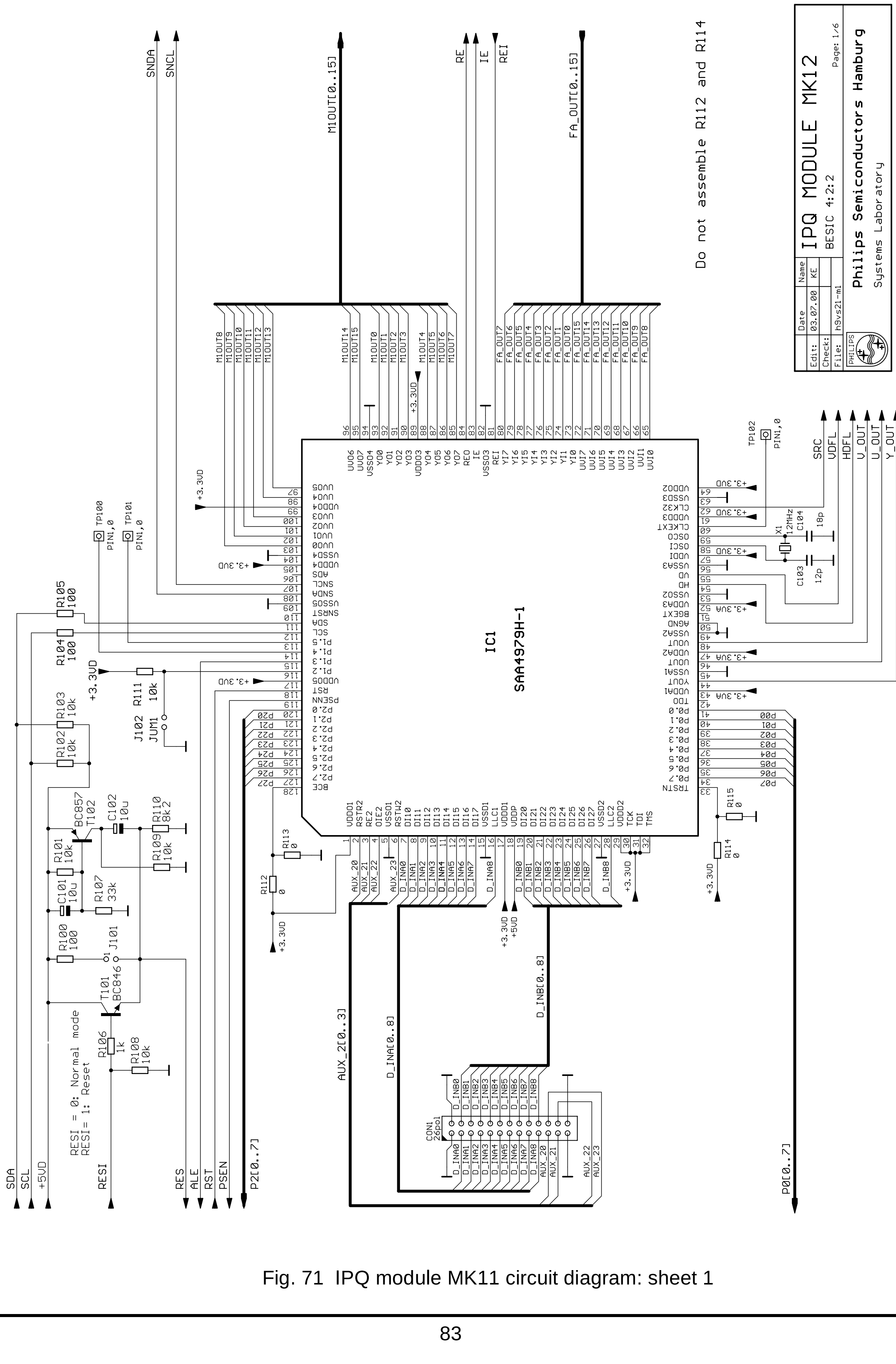


Fig. 71 IPQ module MK11 circuit diagram: sheet 1

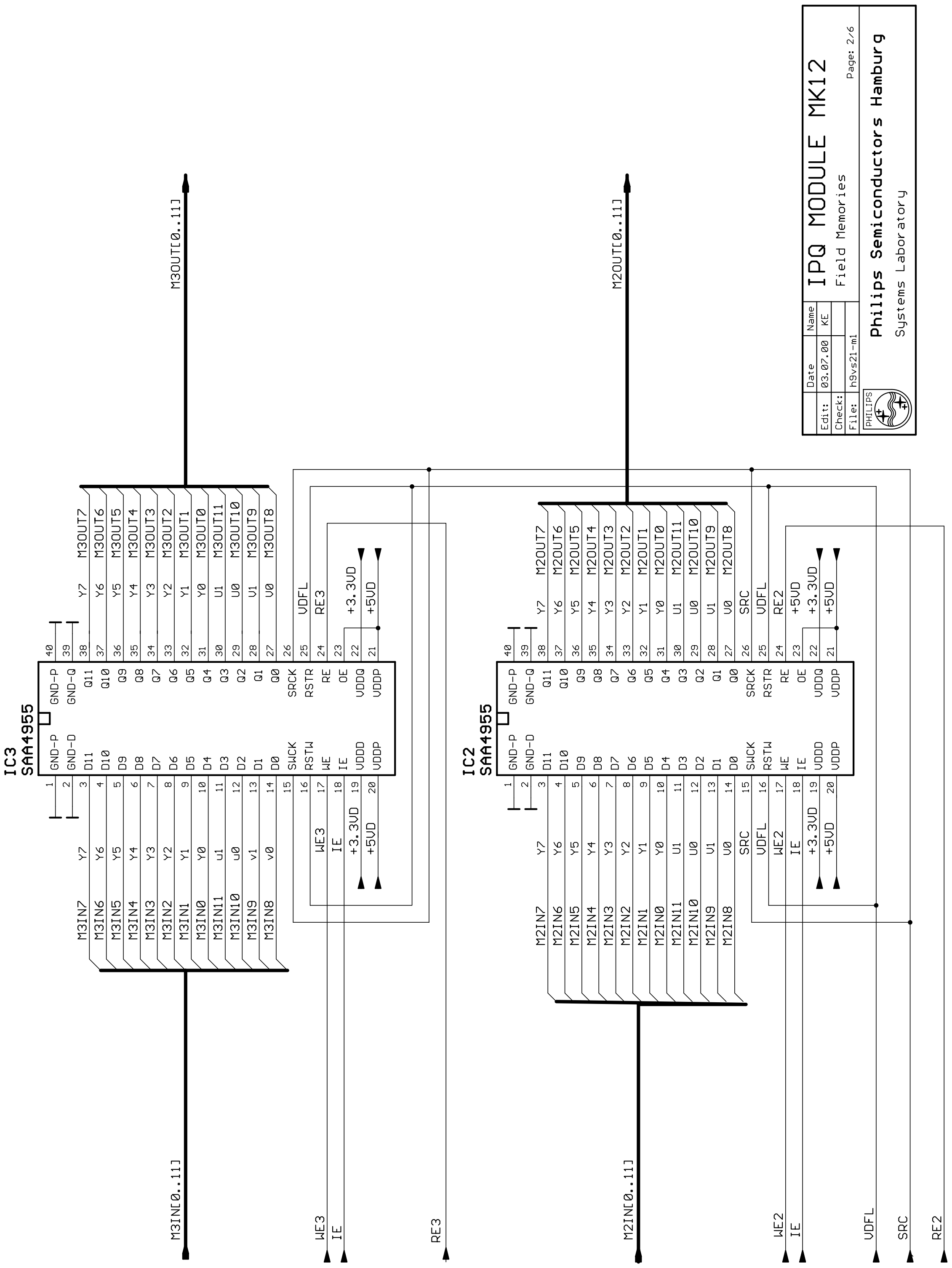


Fig. 72 IPQ module MK12 circuit diagram, sheet 2

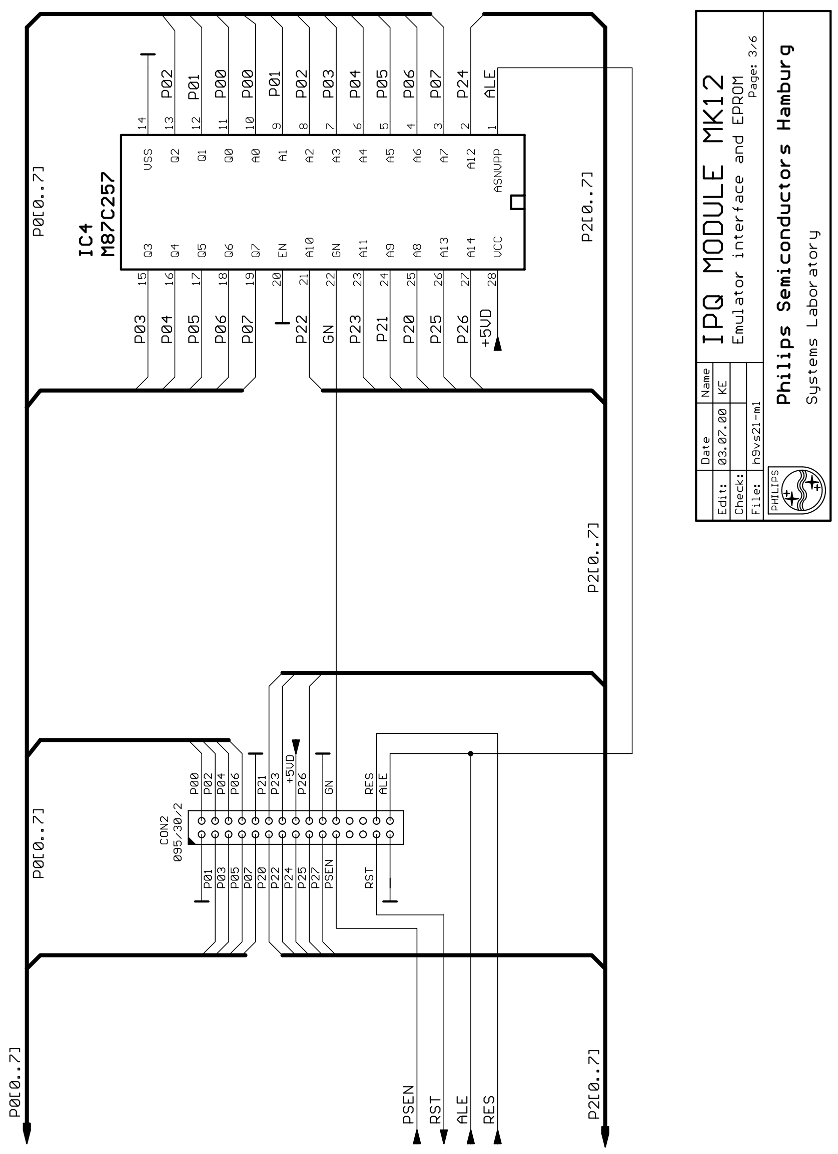


Fig. 73 IPQ module MK12 circuit diagram: sheet 3

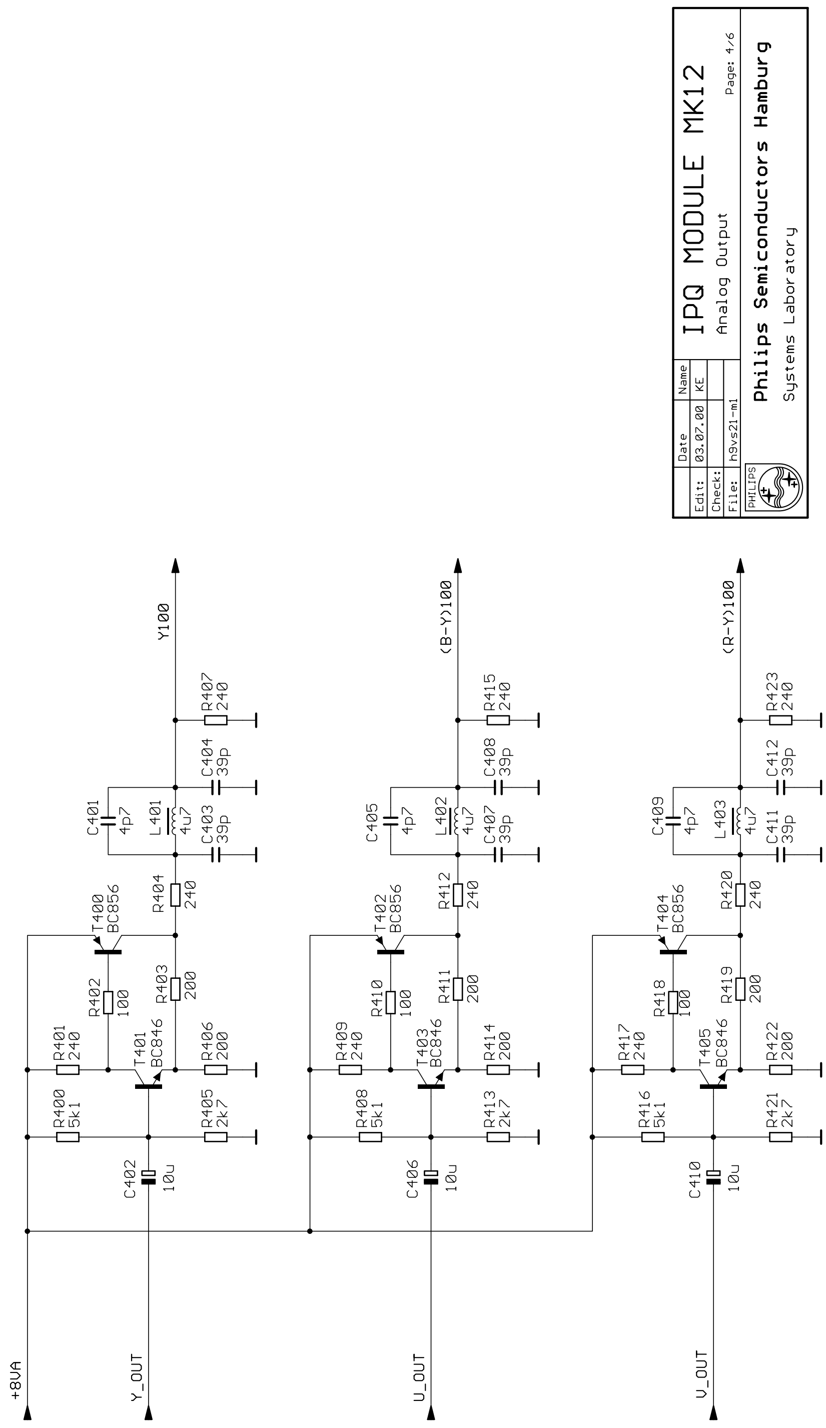


Fig. 74 IPQ module MK12 circuit diagram: sheet 4

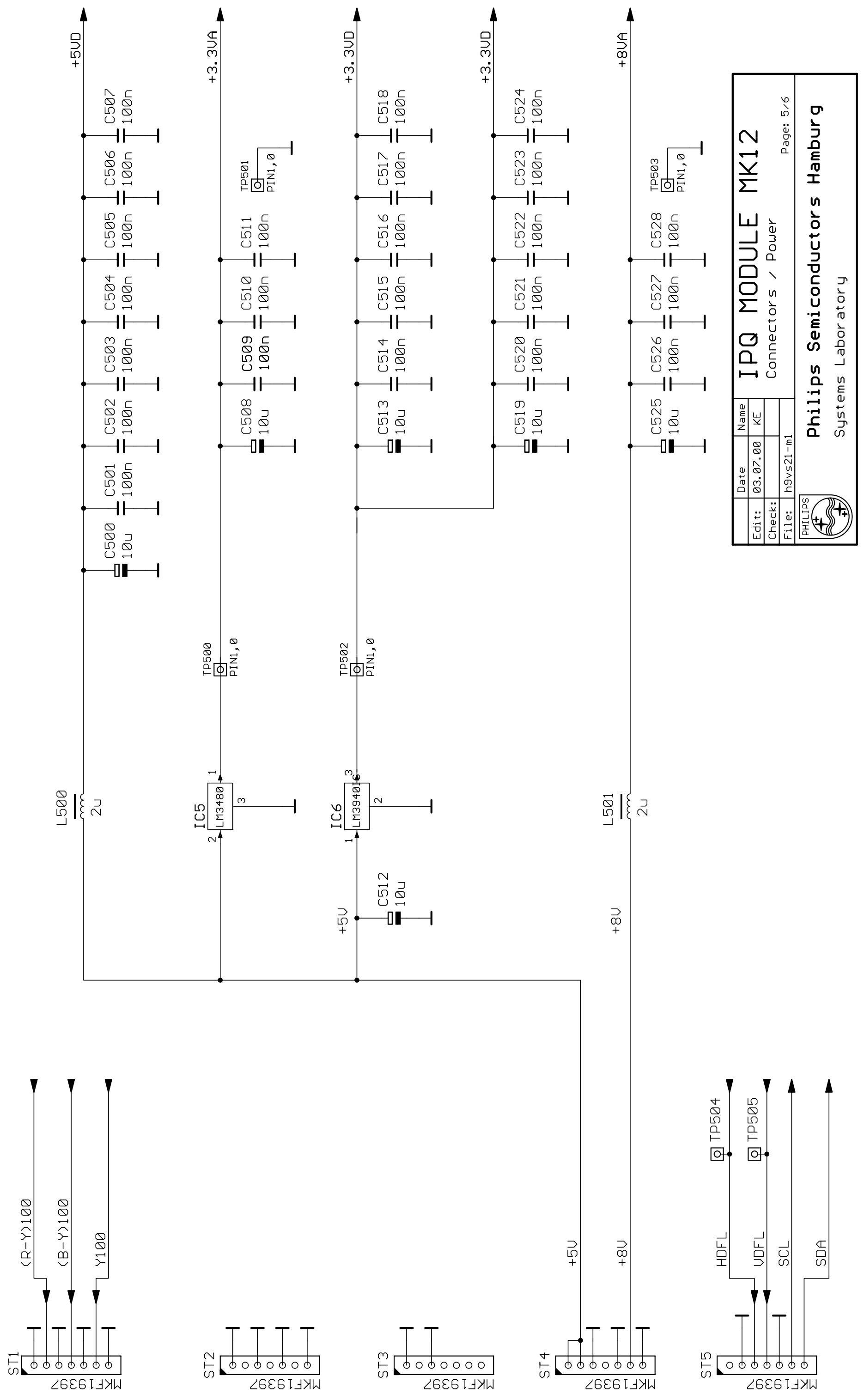


Fig. 75 IPQ module MK12 circuit diagram: sheet 5

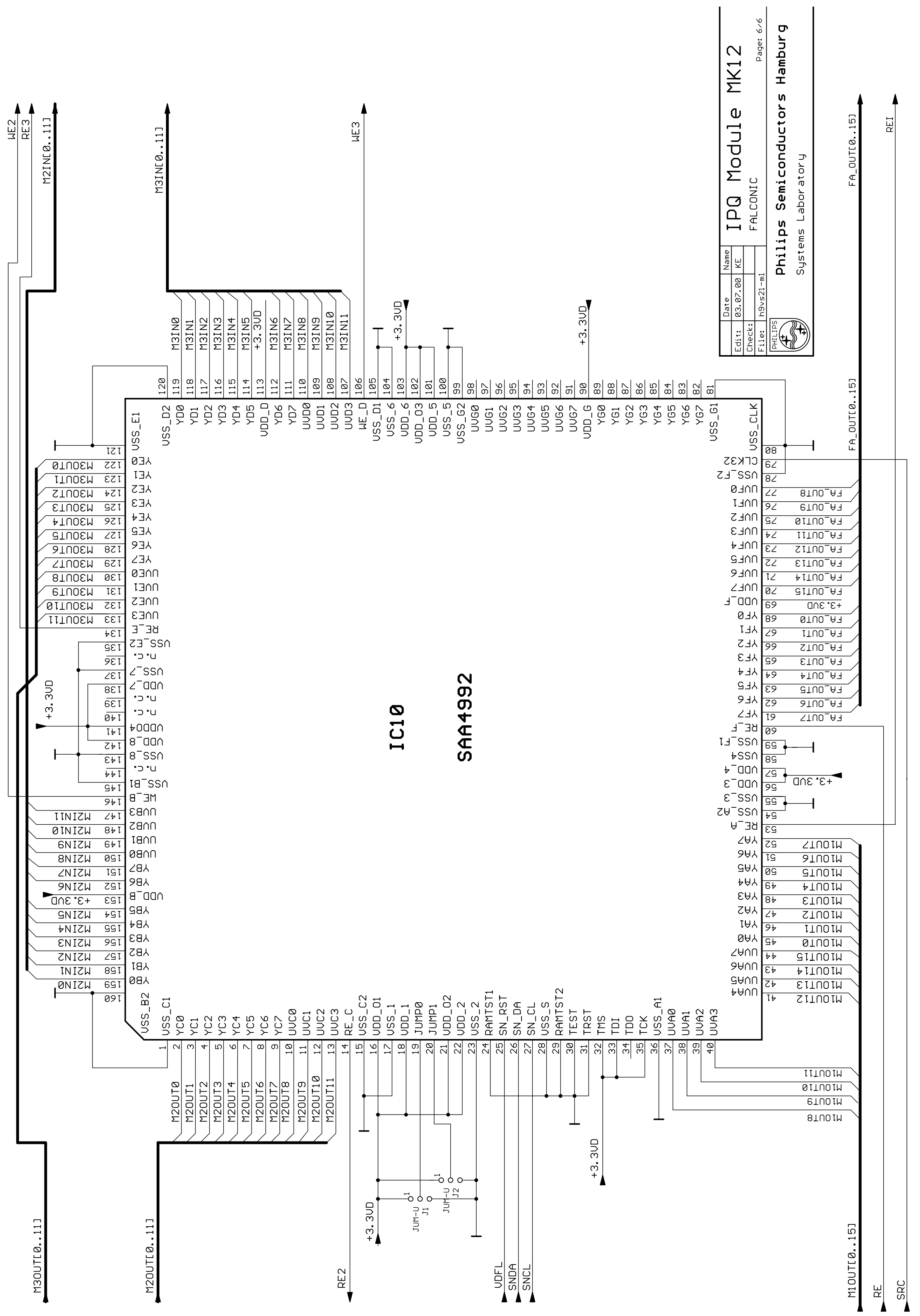


Fig. 76 IPQ module MK12 circuit diagram: sheet 6

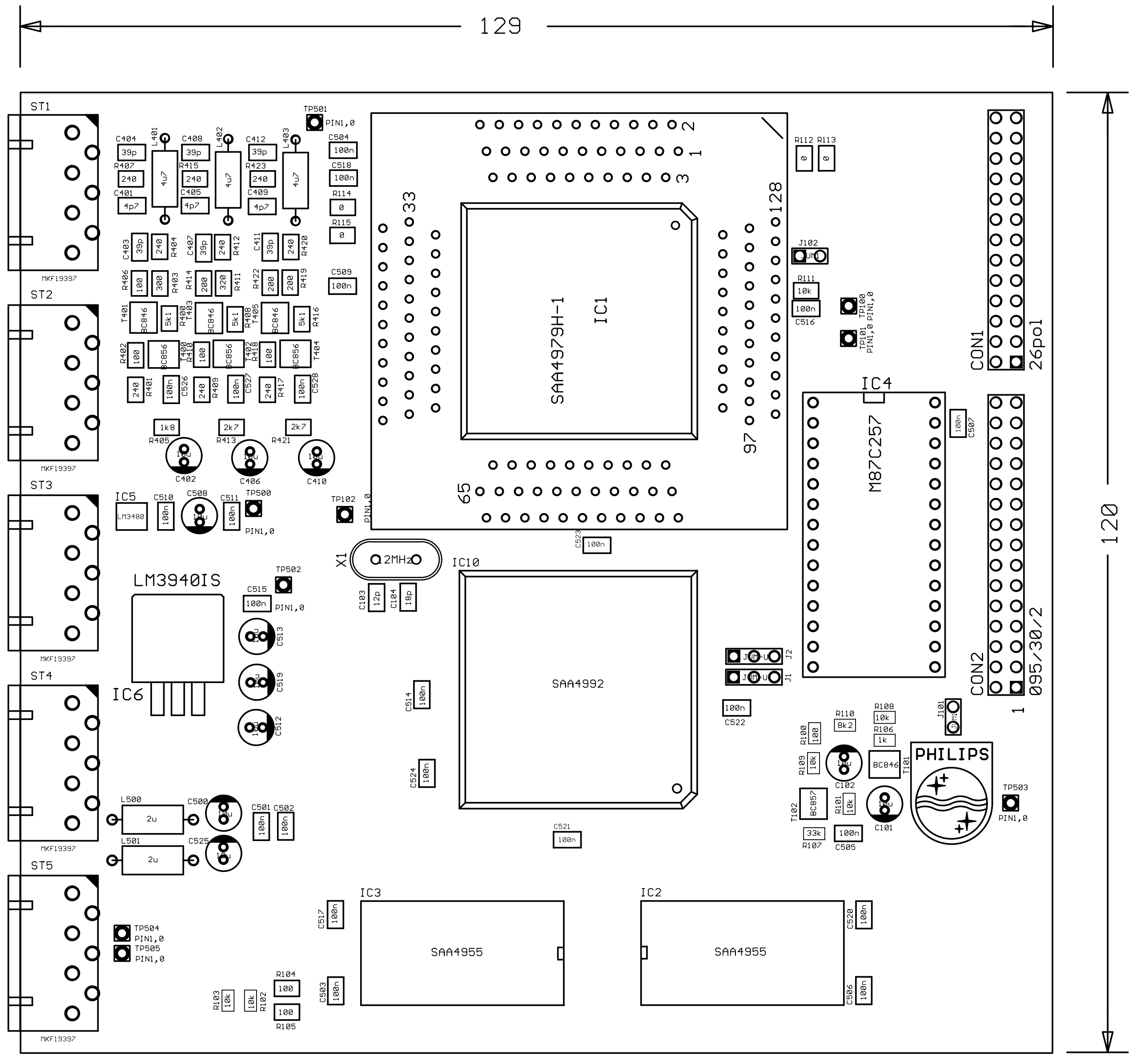


Fig. 77 IPQ module MK12 board layout: position of parts
(layout has provisions for a socket for the SAA4979)

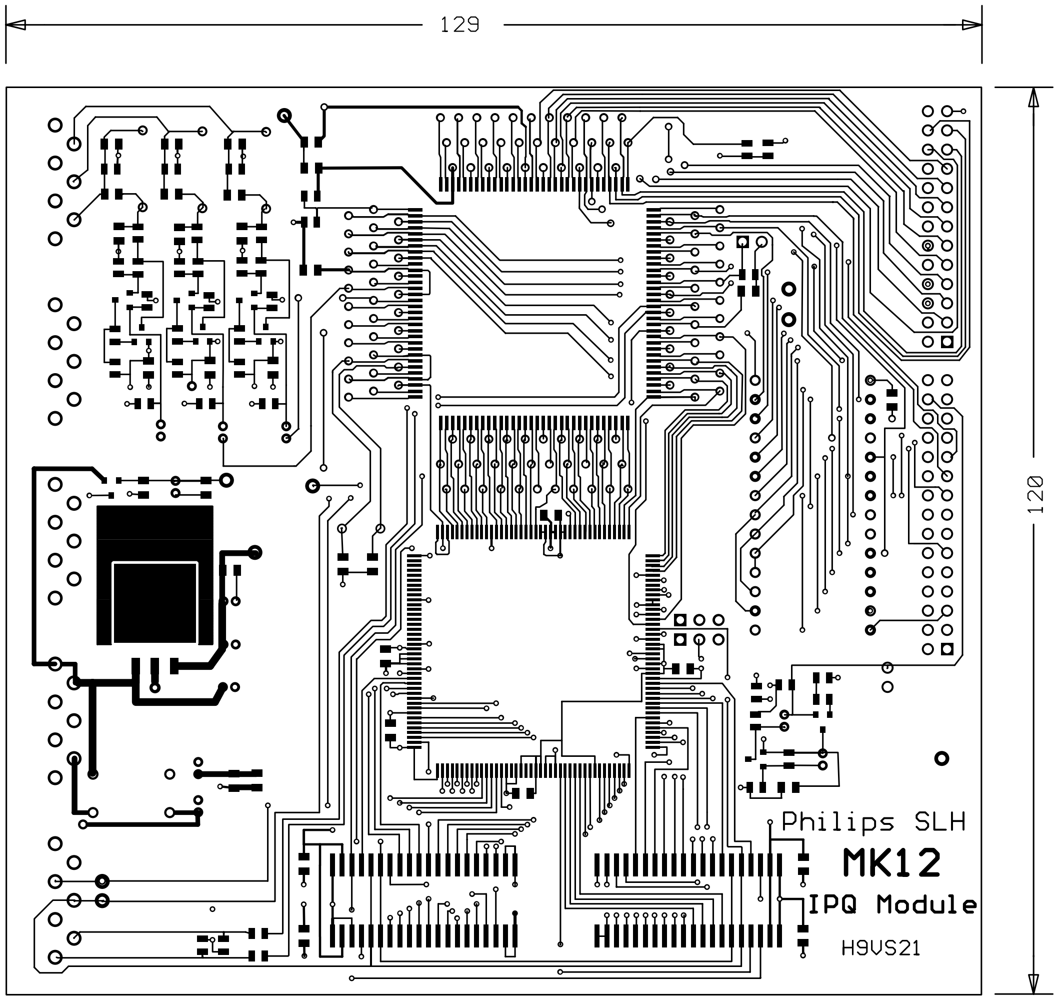


Fig. 78 IPQ module MK12 board layout: layer 1 (top)

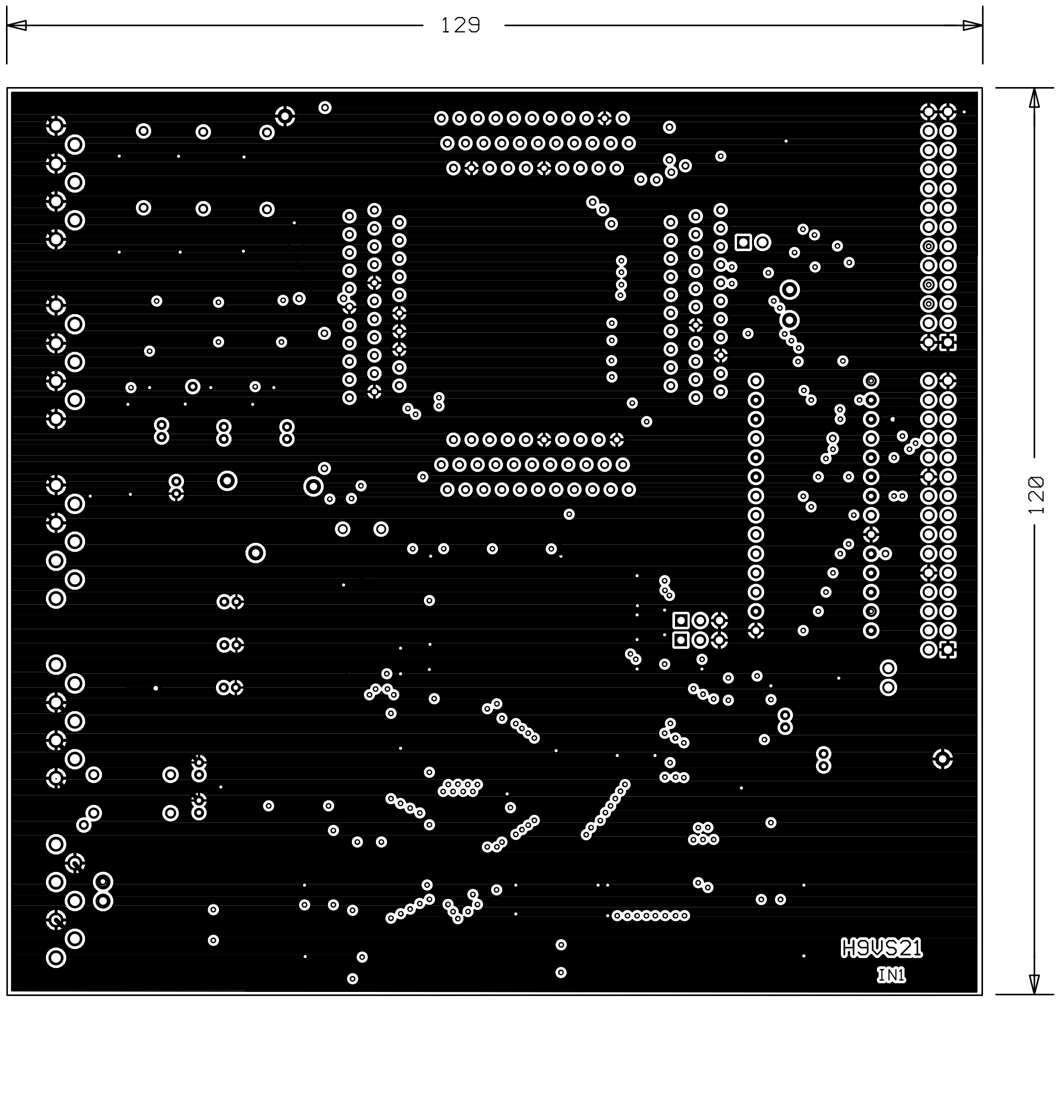


Fig. 79 IPQ module MK12 board layout: layer 2

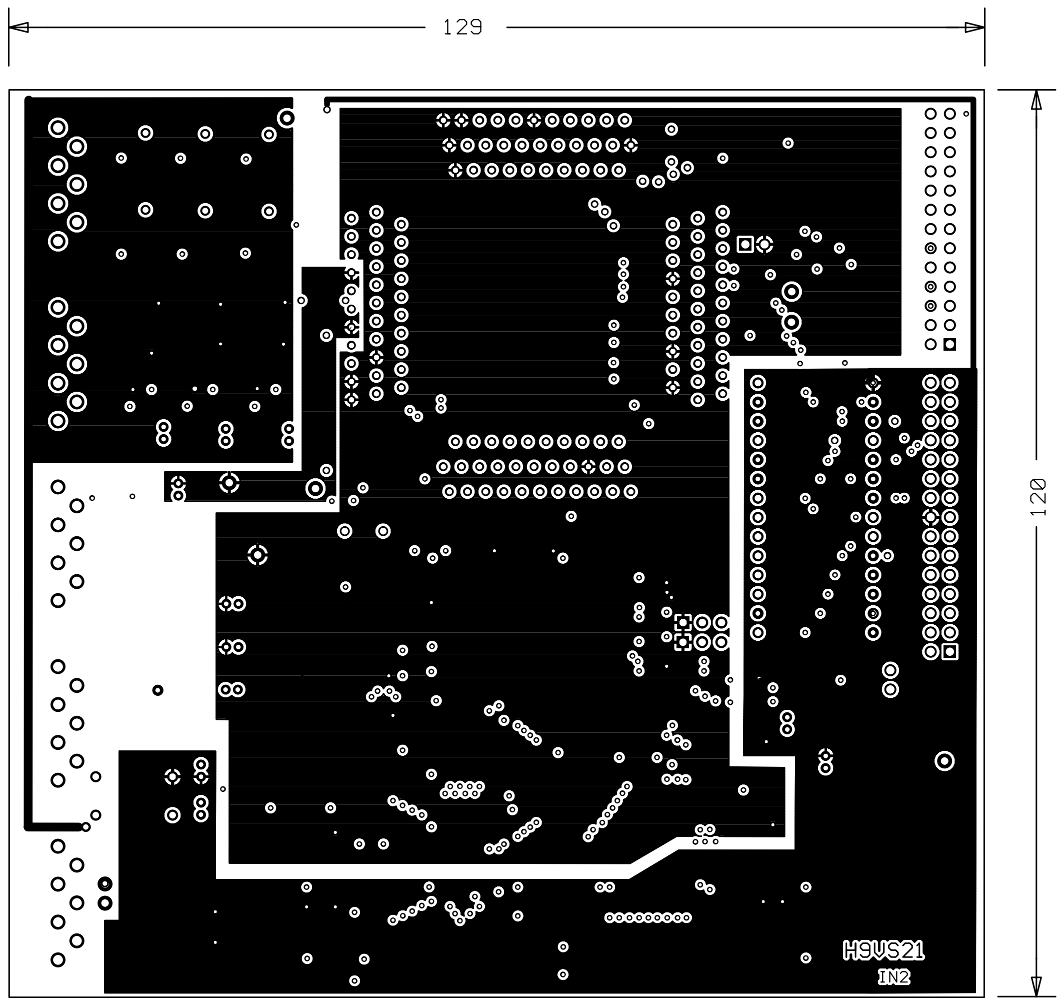


Fig. 80 IPQ module MK12 board layout: layer 3

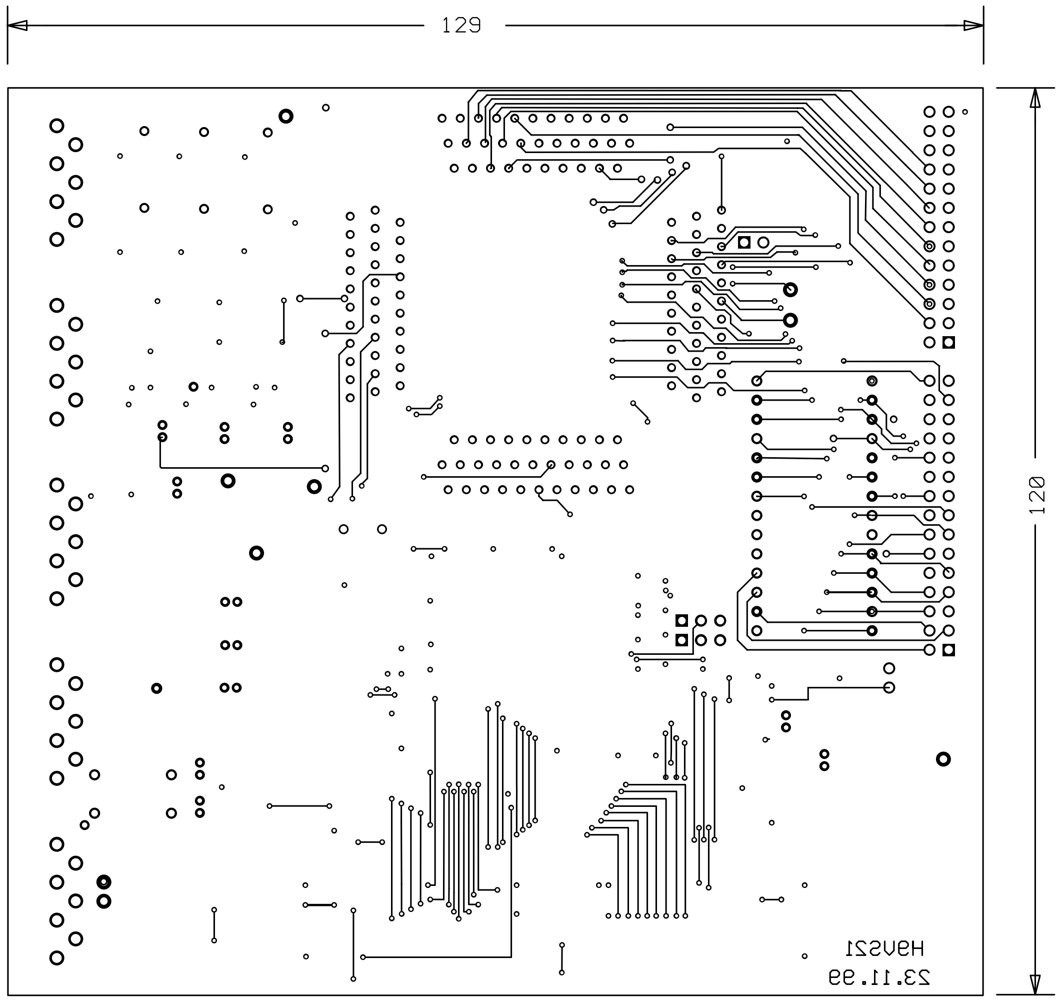


Fig. 81 IPQ module MK11 board layout: layer 4 (bottom)