LM9831 Software Designers Guide

National Semiconductor

Revision 1.0

October 1999

Table of Contents

1. Introduction

The purpose of this document is to ease the design task of the scanner designer. The LM983X family of scanner ICs is complex and powerful. A good first step to utilizing the LM983X devices is to review the data sheet and become familiar with the analog and digital functional blocks of the ‘Scanner on a Chip’ product.

This document is divided into 5 main sections. First is a flowchart which shows the normal flow of events that occur during a scanning operation. This flowchart refers to other parts of the document. These include; a table with the major software elements called during operation, a table listing scanner system performance limits, a section which lists the numerous equations used to calculate system parameters, a section describing typical scanner system parameters and how they are used to determine the LM9831 register settings, and a final section incorporating code fragments.

In this document, references are made to LM983X to refer to the ‘Scanner On a Chip’ family of products from National Semiconductor. Where applicable, differences between LM9830 and LM9831 operation have been described. In Revision 1.0 of this document all register descriptions and settings refer specifically to the LM9831.

2. LM983X Scanner Operation

The following flowchart describes the basic operational elements of a typical scan. The flowchart assumes certain scanner settings have been predetermined in the '.ini' file and selected by the user in the software interface if applicable. Key elements of this chart are described in more detail in later sections.

Figure 1: Basic Scan Flowchart (for CCD Sensors)

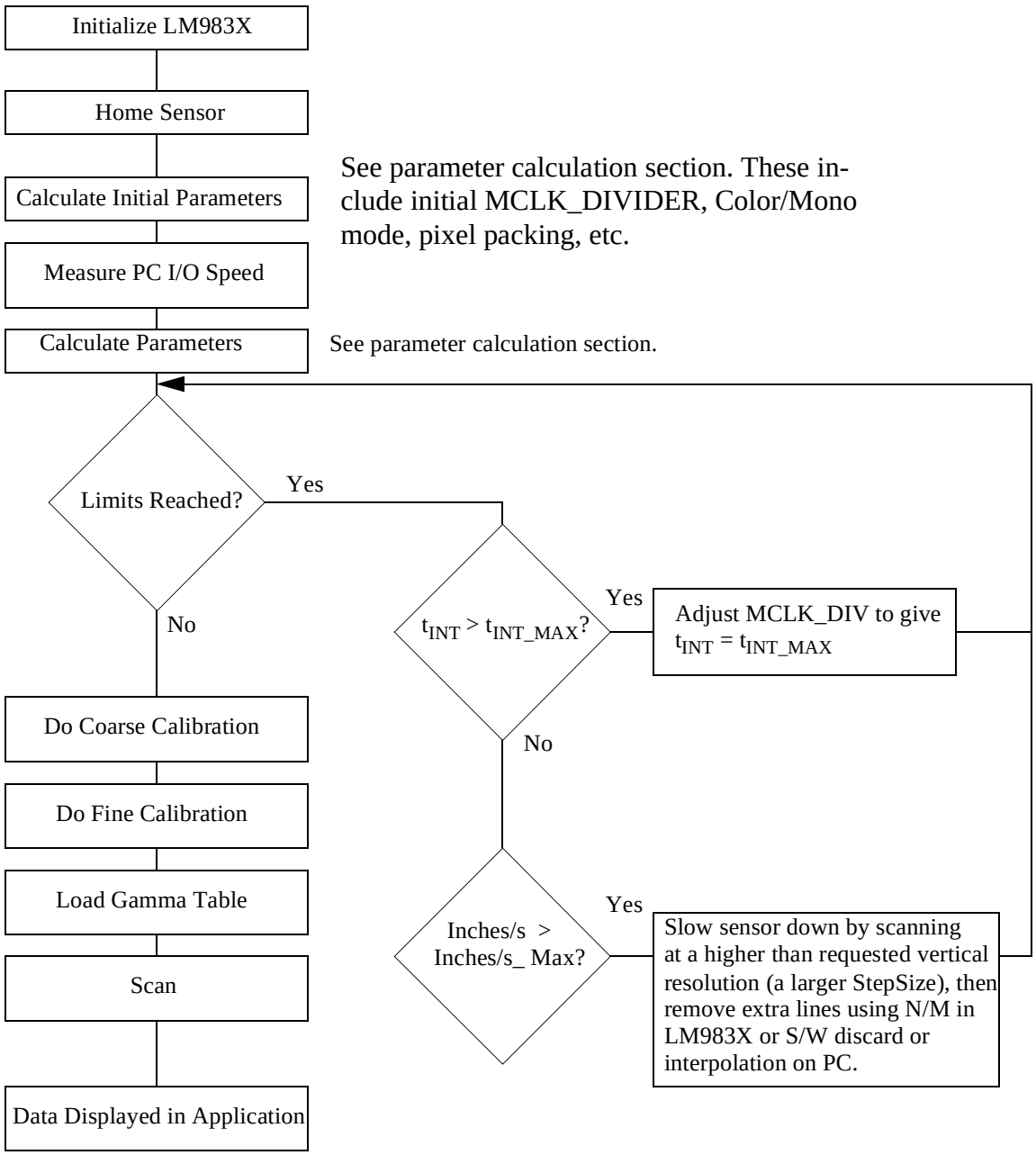


Table 1: Function Descriptions - Related Software

Function Performed	Function Explanation (What is done, why?)	Function Details	Related Software Modules
Scan Button Pressed			
Port Initialization	Ensure communications with target scanner		Startup() in Hardware.cpp
Calculate scanner to PC transfer Rate	Scan data for 1 second without moving sensor or paper. Scan at 8 bits depth, use monochrome or color mode as selected by user. Calculate nominal data transfer rate. This transfer rate is then used to pick the best scan speed parameters to maximize bandwidth over the data connection and minimize pausing during scanning.		SetupScan() in ScanStateMachine.cpp ComputePCTransferRate() in Scan.cpp

Table 1: Function Descriptions - Related Software

Function Performed	Function Explanation (What is done, why?)	Function Details	Related Software Modules
Coarse Calibration	Scan reference image pixels to determine optimum coarse gain and offset settings for R,G,B. (Analog gain and offset prior to ADC) These coefficients are applied at the line rate during normal scanning. LM9831 calibration is done at the horizontal resolution set by the HDPI divider. LM9830 calibration is done at the optical resolution of sensor. The color mode is either color or gray scale as selected. 8 bits per pixel data is used.	Set register 09 = 00111NNN (16 bit datamode, NNN = resolution to scan at) Set registers 3E/3F = 0x0000 (pixel rate offset = 0) Set registers 40/41 = 0x0100 (shading gain = 16384/16384=1) Set register 42 = 0N100000 (Get gain/offset data from registers, not DRAM, N=DRAM size) Set register 45 = XXX0XXXX (disable motor) Loop 1: Scan a calibration strip. The line should contain a white strip with some black at the beginning. [Do the following for each color channel if applicable] Examine the data and calculate min_pixel and max_pixel. Min_pixel corresponds to the minimum black value. Max_pixel corresponds to the maximum white value. Adjust the Static Offset (registers 38-3A) and Static Gain (registers 3B-3D). The goal is to get: 2048 > Min_Pixel > 0 TargetCode > Max_Pixel > TargetCode - 8000 If these two conditions are not met, then goto Loop 1: If the conditions are met, then goto Loop 2 (Fine calibration).	SetupScan() in ScanStateMachine.cpp DoCalibration() in Scan.cpp

Table 1: Function Descriptions - Related Software

Function Performed	Function Explanation (What is done, why?)	Function Details	Related Software Modules
Fine Calibration	Scan reference image pixels to determine the optimum fine gain and offset coefficients. These are calculated on a pixel by pixel basis across the good pixel area of the sensor. This gain and offset correction is done in the LM983X digital section (Immediately after the ADC in the LM9830 and after the ADC and Horizontal DPI adjust in the LM9831). LM9831 calibration is done at the horizontal resolution set by the HDPI divider, LM9830 calibration is done at the optical resolution of the sensor. The color mode is either color or gray scale as selected. 10 bits per pixel data is used for the LM9830 and 14 bits per pixel data is used for the LM9831.	Set register 45 = XXX1XXXX (enable motor) If there is a black calibration strip for all pixels then (the current demo scanner doesn't have this and the current software doesn't support it): Move the sensor to the Black Calibration line. [Do the following for each color channel if applicable] Digitize the black line (you can optionally scan and average multiple lines to reduce noise) Write the black line data to Offset Coefficient data in DRAM. Else (no black calibration strip) Write Min_Pixel value from coarse calibration to Offset Coefficient data in DRAM (write the same Min_Pixel value to the Offset Coefficient value for every pixel in the line) End If Move the sensor to the White Calibration Strip Set registers 4A/4B = 0x0000 (don't skip any lines) Set register 50 = 0x00 (do not reverse when buffer full) Set register 51 = 0x00 (no acceleration profile) Set register 42 = 0N100100 (Turn on Pixel Rate Offset, N = DRAM size) (Continued below)	SetupScan() in ScanStateMachine.cpp DoCalibration() in Scan.cpp

Table 1: Function Descriptions - Related Software

Function Performed	Function Explanation (What is done, why?)	Function Details	Related Software Modules
Fine Calibration (cont.)		Start scanning white calibration strip Register 7 = 3 (start scan) Poll register 3 until scanner is paused (bit 4 = 1) Read x lines of data Reset LM9831 (register 7 = 0x20 then register 7 = 0x00) Repeat y times until all of the calibration area has been scanned Average x * y lines together to reduce noise Calculate Gain Coefficient for each pixel. Gain Coefficient(n) = (Target Code/Pixel_Data(n)) * 16384 Write Gain Coefficient Data to DRAM Set Register 42 = 0N100110 (turn on pixel rate offset and shading correction, N = DRAM size) Fine calibration is complete.	
Home Sensor			
Scan	Image data is now acquired. Parameters written to LM983X registers and gamma, gain and offset coefficients written to RAM. After scanning is complete, data is transferred to the pre-view file, TWAIN application, or file as selected. Scan of image data done at user set resolutions, color depth and window size/location. Scanning is complete when the requested # of lines have been transmitted back to the PC.		ScanSome() in Scan.cpp
Home Sensor			

3. Scanner Performance Limits

Certain key system parameters will limit the performance of any scanner. In the following table we have listed these limits and the potential solutions the LM983X products provide.

Table 2: Scanner System Performance Limits

Resolution	Line Size	PC I/O Speed	**Limit Reached**/Discussion	Potential Solutions
High	Full	Fast	**Integration Time** Limited by DRAM speed in LM9831.	LM9831 use Int_Time_Adj, Reduce lamp intensity
High	Full	Slow	**Integration Time** Limited by DRAM speed in LM9831. **Data generation rate >> than PC I/O rate.** Pausing required to meet constraints.	LM9831 use Int_Time_Adj, Reduce lamp intensity LM9830/31 use N/M to reduce lines transferred. this reduces the average data generation rate.
High	Partial	Fast or Slow	**Integration Time** Limited by DRAM speed in LM9831.	LM9831 use Int_Time_Adj, Reduce lamp intensity
High	Partial	Fast	**Motor speed** limits scan speed Maximum ADC speed limits data generation rate	Scan as fast as mechanics support. Scan at maximum ADC speed.
Low	Full	Either	**Motor too slow**, Int time too long for CCD or if Integration time is OK, motor can't go fast enough.	Set integration time to maximum, increase VDPI setting to scan at higher vertical resolution. Then discard extra lines with N/M in LM983X or discard or interpolate in S/W on PC. Reduce lamp intensity
Low	Partial	Either	**System architecture**. When scanning narrow images, which generate a small amount of data for many lines. May scan to end of document before pause limit is hit. Sensor assembly may run into end of travel in scanner.	Scan a wider block to generate a reasonable amount of data to allow software control of end of scan. Set smallest pause limit possible.

4. Fundamental Scanner Equations

Equation 1: **Stepsize** (pixels/microstep):

$$\text{StepSize} = \frac{\text{VDPI} \times \text{LineLength} \times \text{LineRateColor}}{4 \times \text{FSPI}}$$

Equation 2: **Bytes Per Line** (Approximate number of bytes of data/line assuming 8 bits intensity)

$$\text{BytesPerLine} = \frac{(\text{DataPixelsEnd} - \text{DataPixelsStart})}{\text{HDPI\_ADJ} \times \text{PP}} \times \text{CM}$$

Equation 3: **Pixel Period** (seconds/pixel)

$$\text{PixelPeriod} = \frac{\text{MCLK\_DIV} \times 8 \times \text{CM}}{48\text{MHz}}$$

Equation 4: **Integration Time** (seconds)

$$t_{INT} = \text{PixelPeriod} \times \text{LineLength}$$

$$t_{INT} = \frac{\text{MCLK\_DIV} \times 8 \times \text{CM} \times \text{LineLength}}{48\text{MHz}}$$

Equation 5: **Bytes Per Second** (Approximate amount of data generated in bytes/s)

$$\text{BytesPerSecond} = \frac{1}{t_{INT}} \times \text{BytesPerLine}$$

$$\text{BytesPerSecond} = \frac{\text{CM}}{t_{INT}} \times \frac{(\text{DataPixelsEnd} - \text{DataPixelsStart})}{\text{HDPI\_ADJ} \times \text{PP}}$$

$$\text{BytesPerSecond} = \frac{48\text{MHz}}{\text{MCLK\_DIV} \times 8 \times \text{LineLength}} \times \frac{(\text{DataPixelsEnd} - \text{DataPixelsStart})}{\text{HDPI\_ADJ} \times \text{PP}}$$

Equation 6: **LineDataSize (Bytes)** Exact line data size for LM9831

$$\text{LineDataSize} = INT\left(\frac{INT\left(\frac{\text{DataPixelsEnd} - \text{DataPixelsStart}}{\text{HDPI\_ADJ}}\right) \times \text{CM} \times \text{BitsPerPixel}}{16}\right)$$

Equation 7: **Ideal MCLK Divider (unitless)**

$$\text{MCLK\_DIV} = \frac{48\text{MHz}}{\text{HostIORate} \times 8 \times \text{LineLength}} \times \frac{(\text{DataPixelsEnd} - \text{DataPixelsStart})}{\text{HDPI\_ADJ} \times \text{PP}}$$

Equation 8: **Ideal Scan Speed** (inches/second)

$$\text{MicroStepsPerSecond} = \frac{1}{\text{PixelPeriod} \times \text{StepSize}}$$

$$\text{ScanSpeed} = \frac{\text{MicroStepsPerSecond}}{\text{MicroStepsPerInch}}$$

$$\text{ScanSpeed} = \frac{1}{\text{PixelPeriod}} \times \frac{1}{\text{FSPI} \times 4} \times \frac{1}{\text{StepSize}}$$

$$\text{ScanSpeed} = \frac{48\text{MHz}}{\text{MCLK\_DIV} \times 8 \times \text{CM}} \times \frac{1}{\text{FSPI} \times 4} \times \frac{4 \times \text{FSPI}}{\text{VDPI} \times \text{LineLength} \times \text{LineRateColor}}$$

$$\text{ScanSpeed} = \frac{48\text{MHz}}{\text{MCLK\_DIV} \times 8 \times \text{CM} \times \text{VDPI} \times \text{LineLength} \times \text{LineRateColor}}$$

Symbol Definitions

BitsPerPixel - bits/pixel. Related to inverse of pixel packing. How many bits of data per Gray or Single color pixel. 1, 2, 4, or 8 are valid sizes.

BytesPerSecond - Bytes/Second. Number of bytes of image data generated per second, includes effects of HDPI adjust and PP (pixel packing)

CM - color Mode, unitless. For 3 channel pixel rate color CM=3, for line rate color or gray scale scanning, CM=1

DataPixelsEnd - Pixels. Last valid data pixel in a line

DataPixelsStart - Pixels. First valid data pixel in a line

FSPI - Full Steps/Inch, how many motor full steps are required to move one inch down the page

HDPI_ADJ - Horizontal Dot Per Inch Adjustment unitless, used to reduce horizontal scanning resolution

LineLength - Pixels. Number of pixels between TR (CCD or CIS TRansfer) pulses. See the DPD calculation in “Code Fragments” on page 25 for the exact calculation of LineLength.

LineRateColor - Unitless, equals 3 for line rate color scanning. Equals 1 for pixel rate color and gray scale scanning. Corrects vertical scanning speed for line rate scanning where 3X pixels occur during a single “line” of data at vertical resolution chosen.

MCLK_DIV - Unitless, Master CLocK DIVider. Used to match the scanning and pixel processing rate to the nominal PC data rate, and scanner hardware requirements

MicroStepsPerSecond - Microsteps/Second. Number of microsteps per second

PP - Pixels/Byte. Pixel Packing, as color depth of each pixel is reduced, data from multiple pixels can be packed in a single byte of data. For example for line art mode with 1 bit per pixel in B/W, 8 pixels can be packed in a single byte. 1, 2, 4, 8 are valid settings.

PixelPeriod - Seconds/Pixel. Time for one pixel to be clocked out of the CCD in seconds

PixelsPerLine - Pixels/Line. Adjusted number of pixels per line including effects of HDPI adjust and PP (pixel packing)

StepsPerSecond - Fullsteps/Second. Number of full steps per second or MicroStepsPerSecond/4

StepSize - Pixels/Microstep. Number of pixels per microstep of stepper motor

t_{INT} - Integration time which is the time between two TR pulses

VDPI - Vertical Dot Per Inch, Dots/Inch or Pixels/Inch, describes vertical scanning resolution

5. Parameter Calculation

As we see from the equations section of the document, there are a significant number of parameters that define and affect the operation of the scanner. They control the scanner functionality and performance through the various LM983X register settings. Some parameters are directly related to particular register settings, while others are used in various calculations and more indirectly affect the register values. The parameters can be broken down into several different groups:

5.1 System/Hardware Parameters

These are parameters that are defined by the scanner hardware and do not change during scanner use. They are frequently listed in an initialization file so values can be changed without editing the system software.

Table 3: System/Hardware Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please see the LM9831 data sheet for more information, especially for Direct register parameters.)
Crystal Frequency	Indirect	08,
RAM Pixel Data Buffer Size	Indirect	01,42,4E,4F
Minimum Pixel Data Buffer Limit	Indirect	01
Motor Full Steps Per Inch	Indirect	43-55
Scan Bar Maximum Speed	Indirect	46-49
Motor full steps from home position to start of scanning area	Indirect	4A,4B
Height of calibration strip in motor full steps	Indirect	4A,4B
Width of scan area in pixels	Direct	22-25
Length of scan area in inches	Indirect	n/a
Sensor total number of pixels	Direct	1E-23
Sensor image pixels start and end	Direct	22-25
Sensor CDS on or off	Direct	0B
Sensor signal polarity positive or negative	Direct	0B
Sensor maximum integration time	Indirect	08,19,
Sensor line separation	Indirect	n/a
Sensor standard or even/odd output	Direct	0B
Sensor/System optical resolution	Indirect	0B
Sensor control signals polarity	Direct	0C
Sensor control signals active/disabled	Direct	0D

Table 3: System/Hardware Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please see the LM9831 data sheet for more information, especially for Direct register parameters.)
Sensor control signal pixel rate timing	Direct	0E-18
Sensor control signal line rate timing/modes	Direct	0D,0E,0B
Sensor black clamp timing	Direct	1C,1D
Sensor CIS specific timing	Direct	19
Sensor Toshiba mode timing	Direct	1A
Sensor color modes	Direct	26,27
Illumination Modes/Timing	Direct/Indirect	29-37
Stepper motor control modes	Direct	45,50,54
Stepper motor control timing	Direct/Indirect	46-57
Stepper motor paper sense modes/timing	Direct	4C,4D
Stepper motor pause/reverse modes/timing	Direct	50,51,54
Misc. I/O modes/settings	Direct	59,5A,5B
Red, Green and Blue target full scale ADC readings for calibration	Indirect	n/a

5.2 User Defined Parameters

These are defined by the user of the scanner system. They are usually set through some form of user interface and will often change during scanning use.

Table 4: User Defined Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please see the LM9831 data sheet for more information, especially for Direct register parameters.)
Horizontal Resolution	Indirect	08,09,19
Vertical Resolution	Indirect	46,47,43,44,54
Horizontal resolution reduction method	Direct	09
Vertical resolution reduction method	Direct/Indirect	46,47 or 43,44,54
Color or Gray Scale	Direct	26,29,2A-37
Bit depth (1,2,4,8,...) bits per pixel/color	Direct	09,Gamma Table(s)

Table 4: User Defined Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please see the LM9831 data sheet for more information, especially for Direct register parameters.)
Tone mapping curve (Gamma))/thresholding	Direct	Gamma Table(s),
Preview or Full scan	Indirect	08,46,47
Size and location of scan window in preview	Direct	22,23,24,25,4A,4B

5.3 Measured Parameters

These are determined by measuring properties of the system.

Table 5: Measured Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please see the LM9831 data sheet for more information, especially for Direct register parameters.)
Host-Scanner datalink bandwidth	Indirect	08
Coarse calibration data	Direct	38-3D
Fine calibration data	Direct	Gain/Offset Tables
Sensor saturation point/maximum integration time	Indirect	08,19

5.4 Calculated Parameters/Register Settings

These are determined through calculation, and are based on System Hardware, User Defined and Measured parameters.
Using the three types of input parameters, a number of calculations are done to determine the scanner settings. Scanner settings will include both LM9831 register settings, and software settings used in processing scan data.

One should note that there are many registers that only need to be set once for all subsequent operations. Other registers will need to be accessed and reconfigured more frequently in response to the various scanning operations performed. To help identify the frequently accessed/updated registers, they are highlighted with a ✷.

✷Register 0 (Pixel Data)

The pixel (image) data is read from this register

✷Register 1 (Pixel Data Buffer)

This register tells you how much image data is in the DRAM buffer. The 8 bit value read is interpreted as follows:
(n * 2) kbytes of data for a 256k * 16 DRAM, or

(n * 8) kbytes of data for a 1M * 16 DRAM.

✶Register 2 (Paper Sense and Misc I/O Status)

This register displays the status of the Misc I/O and Paper Sensor pins. The Misc I/O pins can be used for buttons, LEDs, etc. The Paper Sensor inputs can be used to detect the home position of the scanner.

✶Register 3 (DataPort Control)

This is the main control register for the DataPort. The DataPort is used to write and read gamma, shading, and offset data to and from the DRAM. **Data should only be written to or read from the DataPort when Register 07 is set to Idle.** The gain and offset data will be determined during the fine calibration process. Please refer to the calibration section for more information. The gamma data is usually selected by the scanner user through the host application software. Please refer to the LM9831 data sheet for more information

Bit 4 is the read-only Pause bit. It is set if the scanner has entered the pause (buffer full) state.

Bit 6 controls the DRAM Test mode, allowing writing and reading of the entire DRAM for test purposes, and also to detect if a 256k * 16 or a 1M * 16 DRAM is installed in a system.

✶Registers 4 and 5 (DataPort Address and Read/Write Select)

These registers are used for selecting the address when writing to and reading from gain, offset and gamma tables. The actual physical DRAM address is controlled by the LM9831 memory controller, these registers select the address of the data within each table. For Gain and Offset data the address range is 0 to 16383. For Gamma tables, the address range is 0 to 4095. Care should be taken to only select address values within the valid range. Unpredictable results will occur if addresses outside the valid range are entered.

✶Register 6 (Dataport I/O)

This register is where data is written to and read from to access the coefficient tables. The address value in Registers 4 and 5 is automatically incremented whenever one (Gamma Data) or two (Gain/Offset Data) bytes are written to or read from this register.

✶Register 7 (Command Register)

This is the main “Control” register in the LM9831.
Bits 0, 1, and 2 select the operating mode of the scanner. To start and end a scan, home the sensor in a flatbed scanner, or eject the document in a sheet fed scanner. Always make sure these bits are set to the Idle State before issuing a new active command.
Bit 4 can be set to 1 to select Low Power Standby mode, or 0 to select normal operation.
Bit 5 can be set to 1 to cause a soft reset of the LM9831. Caution: Performing a soft reset stops DRAM refresh momentarily and can result in the lost of stored DRAM data including gain, offset and gamma tables.

✶Register 8 (MCLK Divider)

The lower 6 bits of this register control the Master Clock to most of the scanner. The divider used is equal to 1 + <reg 8>/2. To maximize scan speed, this register is generally set so that

scan image data is generated at the same speed that the USB interface can handle (typically 800kbytes/s).

DRAM speed limitations require that the following condition always be met:
MCLK_DIV * HDPI_ADJ * Int_Time_Adj >= 6.

✷Register 9 (HDPI Divider)

This controls the horizontal resolution. It can be programmed to provide resolutions equal to the optical resolution of the sensor divided by 1, 1.5, 2, 3, 4, 6, 8, and 12. For example, to get 200 dpi from a 600dpi optical sensor, the HDPI Divider should be set to 3.

DRAM speed limitations require that the following condition always be met:
MCLK_DIV * HDPI_ADJ * Int_Time_Adj >= 6.

✷Register 9 (Pixel Packing)

This is used to set the number of bits in each pixel after gamma correction. Pixels can be 8, 4, 2, or 1 bit. 8 bit pixels are normally used for color and gray scale scans. Line art mode typically uses 1 bit per pixel (the threshold for line art mode can be set by the gamma table).

✷Register 9 (DataMode)

This controls whether the pixel output data will be 8 bit (or less, determined by the Pixel Packing setting) or 14 bit. 14 bit mode is used for calibration.

Registers 0B through 18

These are used to select the proper sensor configuration and control signal timing. These settings, in combination with the MCLK Divider determine the timing and frequency of the sensor control signals such as PHI1, RS, CP, etc. Please refer to the LM9831 data sheet for details regarding these settings.
Note that when setting the reference and signal sample points (Registers 17,18) the values must differ by at least 2 MCLKs or no data will be generated. Also the range of values for the settings in Registers 0F through 18 depends on the AFE mode and subsequent rate of data coming from the sensor. For Pixel Rate Modes the range is 0 to 23. For Line Rate Modes, the range is 0 to 7.

✷Register 19 (Integration Time Adjust)

Due to DRAM speed limitations, the maximum speed at which the LM9831 can store pixels is 1MHz. The ADC can run at speeds up to 6MHz, but only when the HDPI divider is set to divide-by-6 or greater, which results in a pixel rate of 1MHz or less.

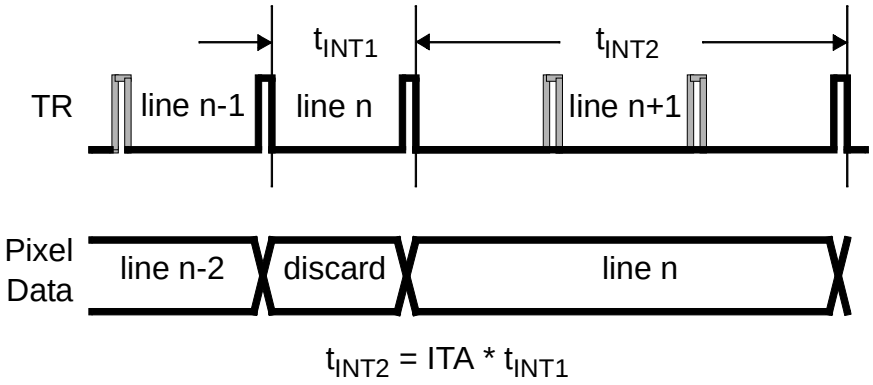
This can be a challenge when scanning at high resolutions. For example, a 600dpi 8.5” wide color CCD scanner digitizes 15,300 pixels/line. At a 1MHz rate, the resulting integration time is15.3ms. Integration times above 10ms may be problematic in some designs.

To allow shorter integration times without violating the 1MHz max pixel rate, the LM9831 has an Integration Time Adjust (ITA) function (Figure 2). ITA generates 2 alternating timebases for the CCD timing, a high frequency timebase, and a lower frequency timebase. During the

high frequency timebase, the integration time (t_{INT1}) is short, as short as the total number of pixels in a line divided by 6MHz. (Using the previous example, that would be 2.5ms). During t_{INT1}, data is clocked out of the CCD but it is not digitized by the AFE. The CCD output signal (representing line "n-1") is discarded.

After the short integration time, the clock is slowed for the next integration time (t_{INT2}). Integration for line "n+1" is done during this period. Since t_{INT2} is longer, there is more time to read out pixel data for line "n". As long as t_{INT2} corresponds to a pixel rate of 1MHz or slower, the line can be digitized and written to the DRAM.

Figure 2: Integration Time Adjust - Two Timebases.



$t_{INT\ 1}$ is determined by the traditional calculations, primarily the MCLK divider and line end settings. t_{INT2} = ITA * t_{INT1}.

There are two more considerations when using the ITA. The first is CCD image lag. Image lag is a sensor phenomenon in which a percentage of the pixel voltage from the previous line appears in the pixel voltage for the current line. In the example above, some of the signal from line n-1 will leak into line n. Since the integration time for line n-1 (t_{INT2}) is 2 to 6 times longer than t_{INT1}, the leakage may be as much as 2 to 6 times the sensor specified image lag. This is usually not a problem. If it is, use a sensor with a low image lag specification, or reduce the brightness of the CCFL light source.

The second consideration is the stepsize calculation. Using the ITA's dual timebases affects the stepsize required to produce an image with the correct vertical resolution. The solution is to calculate the stepsize using the traditional formula, then multiply it by the factor (ITA+1)/ITA:

$$\text{stepsize\_ITA} = \text{stepsize} \cdot \frac{\text{ITA+1}}{\text{ITA}}$$

Registers 1A and 1B (TR to Stepper Phase Correction)

The first step of the scan occurs n pixels (1-16383) after the first TR pulse. This can be used to set the phase timing between the TR pulses and the stepper motor pulses. Presently National

Semiconductor is developing a method to determine the Phase Correction register setting that corresponds to a specific desired phase offset. Until then, this register should be set to 1. Note: A setting of 0 will cause the maximum phase offset and can increase scan time.

Register 1C (Optical Black Pixels Start)

This register should be set to the pixel number of the first optical black pixel in the CCD. This sets the point on the line where clamping occurs to force the external capacitor between the CCD and the LM9831's OS input to it's nominal value. In most CCDs, this register can be set to 0, since the dummy pixels that come between pixel 0 and the first optical black pixel are very similar to the optical black pixels. This will allow more time for clamping.

Register 1D (Optical Black Pixels End)

This register should be set to the pixel number of the last optical black pixel in the CCD. This sets the point on the line where clamping ends.

Registers 1E and 1F (Active Pixels Start)

This 14 bit register determines the pixel where pixel-rate shading and offset correction begins. This will determine the first "visible" pixel of a scan, the first pixel that can see an image placed on the glass. The value of this register is normally determined during calibration. On many calibration strips, there is a black area next to the white calibration strip. The region where the black meets the white determines the first point where image data is available when scanning. Set registers 1E and 1F to this pixel number.

Registers 20 and 21 (Line End)

This 14 bit register should be set to the TOTAL number of pixels in the CCD (including dummy pixels, optical black pixels, active pixels and additional dummy pixels). If this register is set to a number smaller than the number of pixels in the CCD, the CCD may send out erratic image data. The ideal setting is to have: (Line End + TR_Width) is a multiple of (4 * Step-Size). This ensures a constant phase difference between the TR pulses and stepper motor steps.

✷Registers 22 and 23 (Data Pixels Start)

This 14 bit register determines the pixel where image data begins (coordinate x1). This register is always set at the optical resolution of the sensor (i.e. independently of the HDPI divider setting). Image data for pixels < Data Pixels Start is not stored in DRAM.

✷Registers 24 and 25 (Data Pixels End)

This 14 bit register determines the pixel where image data ends (coordinate x2). This register is always set at the optical resolution of the sensor (i.e. independently of the HDPI divider setting). Image data for pixels >= Data Pixels End is not stored in DRAM. This value should be

set to ensure the Integer requirements of the following equation are met without discarding any pixel data.

$$\text{LineDataSize} = INT\left(\frac{INT\left(\frac{\text{DataPixelsEnd} - \text{DataPixelsStart}}{\text{HDPI\_ADJ}}\right) \times \text{CM} \times \text{BitsPerPixel}}{16}\right)$$

✶Register 26 (Color Mode Settings)

This 8 bit register selects a mode of AFE operation for the type of sensor used, and the mode the user chooses (color or monochrome). The settings in this register will generally only be changed when switching from color mode to gray scale mode.

Register 26 and 27 (TR - Transfer pulse settings for 3 Channel Line Rate Mode)

When the LM9831 is configured in 3 Channel Line Rate Mode, part of Register 26 is used to configure the TRred, TRgreen and TRblue timing for different sensors. Register 27 is used for advanced control of TR timing to achieve selective longer integration times for specific color channels in 3 Channel Line Rate mode.

Register 29 (Illumination Mode Settings)

This register selects the lamp drive mode for the specific type of lamp used in the scanner. This setting is done at scanner initialization and would usually not need to be changed.

✶Registers 2A and 2B (Intensity Settings for Illumination Mode 1)

These registers are used to set the PWM duty cycle for CCFL lamp drive circuits. The 11.7 kHz PWM output can be adjusted from a duty cycle of 0 to 100% (0/4095 to 4095/4095). The value in these registers can be used to modulate light intensity for a number of purposes. One key use is to reduce the lamp intensity when longer integration times are required. This reduces the tendency to saturate the sensor at higher resolutions and pixel counts.

✶Registers 2C through 37 (On-Time Settings for Illumination Modes 2 and 3)

These registers select the lamp timing. Each (Red, Green, Blue) channel of LEDs has independent control of the pixel count at which they turn on and turn off. Normally, the on pixel count value should be a lower value than the off pixel count value. To keep the LEDs constantly on, set the on pixel count value in the valid pixel range, and set the off pixel count value to a number greater than the Line End pixel number. To keep the LEDs off, set the off pixel count value inside the valid pixel range, and set the on pixel count value to a number greater than the Line End pixel number.

The “On” time of the lamp will generally be set as high as possible to give lower sensitivity to ambient light with CIS sensors. If long integration times cause sensor saturation, reduced “On” times can be used to prevent saturation.

✶Registers 38 through 3D (Static Offset and Gain Settings for Analog Front End)

These registers set the gain and offset values used in the analog gain and offset correction circuitry in the analog front end of the LM9831. The values will have an initial value which is then adjusted during the coarse calibration procedure. After the coarse calibration procedure, these values will be fixed until the next coarse calibration procedure is performed. Calibration intervals are up to the system designer and can be as often as once per scan, and as infrequent as once per scanning session. Please refer to the coarse calibration procedure for more information.

✶Registers 3E through 41 (Static Pixel Rate Offset and Gain Settings)

These registers set the fixed gain and offset values used in the digital pixel rate gain and offset correction block. The values will have an initial value which is used during the fine calibration procedure. After the fine calibration procedure, the values in the DRAM gain/offset tables will be used. Calibration intervals are up to the system designer and can be as often as once per scan, and as infrequent as once per scanning session. Please refer to the fine calibration procedure for more information.

✶Register 42 (Pixel Rate Offset/Gain Control, DRAM size select)

Bit 0 of this register controls whether the pixel rate gain multiplier is used or bypassed.
Bits 1 and 2 determine if the gain and offset coefficients come from registers 3E to 41, or from the DRAM tables. During fine calibration the register values will be used, after fine calibration the DRAM gain/offset tables will be used.
Bit 6 selects the DRAM size being used. Set to 0 for 256k * 16, set to 1 for 1M * 16.

✶Register 43 (n, Line Skipping)

n lines of data are saved in DRAM for every m lines (register 44) scanned. This function is bypassed if register 43 is set to 0. Value of n = 256 - <reg 43 value>. Register 54 bits 3 to 7 are also used to select additional properties of the ‘n out of m’ function.
This function would be used when the maximum motor speed limits the vertical scan rate. Scanning will be done at a higher resolution and only ‘n out of m’ lines of data are saved to reduce data bandwidth requirements.

✶Register 44 (m, Line Skipping)

n (256 - register 43) lines of data are saved in DRAM for every m lines scanned. This function is bypassed if register 43 is set to 0. If m = 0 this function is bypassed. Register 54 bits 3 to 7 are also used to select additional properties of the ‘n out of m’ function.
This function would be used when the maximum motor speed limits the vertical scan rate. Scanning will be done at a higher resolution and only ‘n out of m’ lines of data are saved to reduce data bandwidth requirements.

Register 45 (Stepper Motor Mode)

Bit 0 of this registers selects Full Step mode or Microstepping mode.

Bit 1 selects single phase or two phase stepping. Two phase output is necessary for microstepping mode.
Bits 2 and 3 select the polarity of the A,$\overline{A}$,B,$\overline{B}$ output signals. Usually these bits are set to 0. With most drive control circuits, settings these bits to 1 will leave unipolar motors energized in the idle state, and can destroy bipolar motor drive circuits by energizing all transistors in the H-bridge circuit.

✶Registers 46 and 47 (Scanning Step Size)

This 16 bit register determines the speed of the stepper motor relative to the integration time of one line. Stepsize is calculated using the following formula:

$$\text{StepSize} = \frac{\text{VDPI} \times \text{LineLength} \times \text{LineRateColor}}{4 \times \text{FSPI}}$$

StepSize (in microsteps) = # of pixel periods between microsteps (registers 46/47, and 48/49)

LineLength = length between TR pulses, measured in pixels. (This is equal to the Line End [registers 20/21] + the additional length of the TR pulse(s). See the DPD equation for an exact calculation of LineLength.)

VDPI = vertical resolution you want to scan (or move) at (600dpi, 150dpi, etc....)

FSPI = Full Steps Per Inch = # fullsteps that the motor needs to move the image sensor 1" with respect to the image

LineRateColor = 3 for line rate color scanning, 1 for pixel rate color and gray scale scanning.

4 = conversion factor to convert full steps per inch to microsteps per inch

When using the Integration Time Adjust function, the stepsize needs to be modified. The stepsize should be multiplied by the factor (<reg 19> + 1)/<reg 19>, where <reg 19> is the content of register 19.

For example, if register 19 = 3, then you need to multiply the stepsize value by 4/3 to get the correct image.

In some cases it is useful to know what the PPS (pulses per second) being sent to the motor is. To calculate that, use the following 2 equations

$$\text{PixelPeriod} = \frac{\text{MCLK\_DIV} \times 8 \times \text{CM}}{48\text{MHz}}$$

and PixelFrequency = 1/PixelPeriod.

PixelFrequency = the rate at which new pixels come out of the CCD (= RS frequency)

8 = # MCLKs per pixel

CM = 1 for line rate, 3 for pixel rate color (see register 26)

MCLK_DIV = 1 + <reg 08>/2

PulsesPerSecond = Steps Per Second = PixelFrequency/StepSize

So to calculate pulses per second, if a “pulse” = 1 fullstep, then:
pps = 48MHz / (8 * CM * MCLK_DIV * 4 * stepsize)

If a “pulse” = 1 microstep, then
pps = 48MHz / (8 * CM * MCLK_DIV * stepsize)

Registers 48,49 (Fast Feed Step Size)

These registers select the stepping rate during fast forward and fast reverse operation. These will be set with a fixed value based on the maximum scan speed that the scanner hardware is capable of. This value is usually determined experimentally during hardware evaluation and a max speed value slightly lower than the hardware maximum is used for the setting in registers 48 and 49. Frequently this value will be entered in ‘.ini’ file settings to allow easy changes as software and hardware are developed independently.

✶Registers 4A,4B (Full Steps to Skip at Start of Scan)

When the scan starts, the scan head (or paper in sheet fed scanners) is driven forward n full steps (n = 0 to 32767) at the Fast Feed Step Size speed. During previews, this is used to move from the home position to the start of the active scan area, skipping past the calibration strip. During scanning, this allows the scanning to fast forward to the scan area selected in the preview window (coordinate y1). The number of full steps is calculated by y1 (start of scan location in inches) multiplied by the vertical resolution of the scanner in full steps per inch (FSPI).

Registers 4C,4D (Full Steps to Scan after PAPER SENSE 2 trips)

This adds a delay of n = 0 to 4095 full steps between when the PAPER SENSE 2 input trips and when the scanning bit is reset, terminating the scan and motor movement. This is useful for stopping the scan in sheet fed scanners where the paper sensor is located earlier in the paper path than the scan head. The value of n is determined by the distance between the paper sensor and the scanning area in full steps and is dependent on the hardware resolution of the drive system in Full Steps Per Inch (FSPI).

Register 4E (Pause Scanning Buffer Limit)

The value in this register determines how much data is stored in the pixel data buffer before scanning is paused. Pause Buffer Limit = n * 2 kbytes (256k * 16 DRAM) or n * 8 kbytes (1M * 16 DRAM). When scanning is paused, the LM9831 continues scanning until the current line is finished, then pauses. Register 4E must be set to allow this final line to finish scanning with-

out overflowing the scan buffer. For optimum performance with minimum pausing, register 4E should be set according to this formula:
PauseThreshold (kbytes) = AvailableMemory - (LineDataSize + 1)
AvailableMemory = 296 kbytes (256k * 16 DRAM) or 1832 kbytes (1M * 16 DRAM)
See Equation 6 on page 9 for details on calculation of LineDataSize.

Register 4F (Resume Scanning Buffer Limit)

The value in this register determines how much data remains in the pixel data buffer before scanning is resumed. Resume Buffer Limit = n * 2 kbytes (256k * 16 DRAM) or n * 8 kbytes (1M * 16 DRAM).

Register 50 (Full Steps to Reverse When Paused)

The value in this register determines how many full steps (0 to 63) the scanner reverses during a pause event. When set to 0 reversing is disabled. Reversing during pause is used to ensure no image data is lost during pauses. The scan mechanism stops, reverses, stops and continues scanning forward when the resume event occurs. Scanning begins again at the exact point (and at the exact speed and stepper phase) it was discontinued during the pause. The number of full steps to reverse during pause is determined by the amount of physical slack in the scanning system. The number of steps to reverse set in register 50 is in addition to the number of steps set in the acceleration profile in register 51.

Register 51 (Acceleration Profile)

This register sets how many full steps are taken at 0%, 25% and 50% of full speed during any start, stop and reversing pause of the stepper motor. The acceleration profile is NOT used during non-reversing pauses. During the reversing process, the motor drive will pause for 0 to 3 full step time units as set in bits 2 and 3 of register 51. When accelerating or de accelerating, 0 to 3 full steps can be taken at 25% and 50% speeds, as set in bits 2,3,4 and 5 of register 51.

✶Registers 51, 52, 53 (Default Phase Difference)

These registers are used to set when the motor resumes after reversing and pausing. The two MSBs of registers 51 are the most significant bits of the 18 bit word. Valid DPD settings are from 1 to 262143. Please refer to “Code Fragments” on page 25 for details on calculating DPD.

Register 54 (Lines to Process After Pause/Lines to Discard After Resume)

The three MSBs of this register set how many lines of data are processed after the pause command and are discarded after the resume command. These settings are only used during non-reversing pauses and intended to minimize vertical spatial distortion in scanners that do not support reversing pause. This setting will typically be determined experimentally and is very dependent on the scanner hardware and scan speed. The effects of this register and the non-reversing pause phenomenon are easily evaluated by scanning thin diagonal lines and observing the spatial distortion during pause events. Register 54 can be adjusted to minimize these effects.

Register 54 (Line Skipping Phase, Line Skipping Color Phase Delay)

Bit 3 is used to select the color phase order for the ‘n out of m’ function.

Bits 4 to 7 are used to select the Color Phase Delay for the ‘n out of m’ function.

Register 55 (Kickstart Steps - Fullstepping Mode)

When fullstepping bits 0 to 2 are used to set the number of full steps (0 to 7) when maximum current (0.465V/Rsense) is applied when starting the motor. After Kickstart steps, the normal current value of 0.325V/Rsense is applied. This setting provides higher startup torque for motors when fullstepping mode is used. This setting is determined experimentally based on the scanner hardware and would frequently be placed in an ‘.ini’ file for ease of adjustment.

Register 55 (Hold Current Timeout)

Bits 4 to 7 set the hold current timeout for the stepper motor. The hold current is applied for 1 to 31 full step time units after the motor stop is issued. This provides holding torque until the mechanical motion in the scanner system has stopped, then de-energizes the stepper motor to reduce power requirements and heat buildup. This value is determined experimentally based on the scanner hardware and would frequently be placed in an ‘.ini’ file for ease of adjustment.

Register 56 (Stepper Motor PWM Frequency)

This register selects the PWM frequency of the stepper motor drive system. The frequency is equal to:
Fpwm = OSC/(256 * n) for 0 < n < 256.
Fpwm = OSC/(256 * 256) for n = 0.
This value is determined experimentally based on the motor inductance and motor drive circuitry. The value is adjusted for optimum motor torque/speed and minimum power consumption. This setting would often be set in an ‘.ini’ file for ease of adjustment during development. A good initial setting for development would be n = 8.

Register 57 (Stepper Motor Minimum PWM Duty Cycle)

This register allows the minimum PWM duty cycle to be set from 0% to 98% (0/64 to 63/64). The LM9831 PWM drive system has a configurable ‘deadtime’, where current sensing is not performed after the drive is turned on. This allows the current sense system to reject the transients that occur during drive turn-on. After the transients have passed, current sensing is enabled to permit the PWM system to control the current delivered to the stepper motor winding. Register 57 should be set to the smallest value possible that still allows the transients to be rejected. Too large a value will prevent proper current sensing and can cause uncontrolled currents to be delivered to the stepper motor. A recommended initial value for register 57 is 10.

Register 58 (Paper Sense Settings)

This register configures PAPER SENSE 1 and PAPER SENSE 2 INPUTS. These two inputs are highly configurable and either one can be used to clear the command register and stop the scan or stop scan motion. PAPER SENSE 1 is frequently used as the home position sensor in flatbed scanners. PAPER SENSE 2 is often used as the end of page detector in sheet fed scanners. The key difference between the two inputs is that PAPER SENSE 2 will stop the scan after the number of lines specified in the “Lines to Scan after PAPER SENSE 2 trips” setting (Registers 4C/4D). The settings for this register will frequently be included in the ‘.ini’ file for ease of adjustment during development.

Registers 59 through 5B (MISC I/O Pin Settings)

These registers are used to configure the 6 MISC I/O pins on the LM9831. MISC I/O pins 1, 2, and 3 default to inputs, while MISC I/O pins 4, 5 and 6 default as outputs after power up. Care should be taken when using the MISC I/O pins in the opposite function as the power up defaults, to prevent damage during user operation before the LM9831 has been configured by software. These pins can be used to sense multiple input switches and pushbuttons, and to illuminate LEDs etc.

6. Code Fragments

DPD Equation

Here is DPD functions from the code. It supports both LM9830 and LM9831 by calling the functions LM9830() and LM9831() to test for the chip type.

```
////////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------
// FUNCTION      :       ComputeDPD(int *regs,CHardware *pHardware,
//                                      int *tr,BOOL bWantCM /*=TRUE*/)

// DESCRIPTION           :       compute dpd reg 52:53

// INPUTS        :       regs     - merlin registers
//                       pHardware - hardware interface object
//                       tr       - place to store calculated tr
//                       bWantCm          - TRUE if calculated tr contains cm component
//                               - FALSE if calculated tr does not include cm component

// RETURNS       :       dpd
//--------------------------------------------------------
// bWantCM = TRUE by default, if bWantCM=0, then tr is returned with cm removed
int CHardware::ComputeDPD(int *regs,int *tr,BOOL bWantCM /*=TRUE*/,BOOL bShowStatus /*=FALSE*/)
{
int linend;      // line end value reg 20:21
int actpst;      // active pixel start reg 1e:1f
int vps;         // toshiba valid pixels reg 1a:1b
int tcdp;        // toshiba cis dummy pixels 1b
int tpspd;       // turbo/preview mode speed reg 0a b2..3
int tpsel;       // turbo/preview mode select reg 0a b0..1
int gbnd;        // guardband duration reg 0e b4..7
int dur;         // pulse duration reg 0e b0..3
int ntr;         // number of tr pulses reg 0d b7
int afeop;       // scan mode, 0=pixel rate, 1=line rate,
                 // 4=1 channel mode a, 5=1 channel mode b, reg 26 b0..2
int ctmode;      // CIS tr timing mode reg 19 b0..1
int qtcnt;       // quarter speed count count reg 51 b2..3
int hfcnt;       // half speed count reg 51 b0..1
int strev;       // steps to reverse reg 50
int st;                  // step size reg 46:47
int dpd;         // calculated dpd reg 52:53
int tp;                  // tpspd or 1 if tpsel=0
int cm = 1;      // 3 if 3 channel line rate or 1 channel mode b
                 // 1 if pixel rate or 1 channel mode a
int afelat = 7;          // 4 if afeop=0, 7 otherwise
int b = 1;       // if ctmode=0, (ntr+1)*((2*gbnd)+dur+1), otherwise 1
int a = 0;       // if vps != 0 tcdp*(linend - actpst - afelat)/vps, otherwise
0
//new following

        // compute linend
        int value = regs[0x21]; // line end lsb
        linend = value;
        value = regs[0x20]; // line end msb
        linend += 256 * value;
        if (bShowStatus)
                NSCStatusOut("linend=%d,%X",linend,linend);

        // compute active pixel start
        value = regs[0x1F]; // active pixel start lsb
        actpst = value;
        value = regs[0x1E]; // active pixel start msb
        actpst += 256 * value;
        if (bShowStatus)
                NSCStatusOut("actpst=%d,%X",actpst,actpst);

        // compute vps
        value = regs[0x1B]; // vps lsb
        vps = value;
        value = regs[0x1A]; // vps msb
        vps += 256 * (int) (value & 1);
        if (bShowStatus)
                NSCStatusOut("vps=%d,%X",vps,vps);

        tcdp = (value & 0x1F)/2; // toshiba cis dummy pixels
        if (bShowStatus)
                NSCStatusOut("tcdp=%d,%X",tcdp,tcdp);

        value = regs[0x0A]; // turbo/preview mode speed
        tpspd = (value & 0xC)/4;
```

```
if (bShowStatus)
        NSCStatusOut("tpspd=%d,%X",tpspd,tpspd);

tpsel = value & 3; // turbo/preview mode select
if (bShowStatus)
        NSCStatusOut("tpsel=%d,%X",tpsel,tpsel);

value = regs[0x0E]; // guardband/duration
gbnd = (value & 0xF0)/16;
if (bShowStatus)
        NSCStatusOut("gbnd=%d,%X",gbnd,gbnd);

dur = (value & 0xF);
if (bShowStatus)
        NSCStatusOut("dur=%d,%X",dur,dur);

value = regs[0x0D]; // number of tr pulses
ntr = value/128;
if (bShowStatus)
        NSCStatusOut("ntr=%d,%X",ntr,ntr);

// afeop = 0 if pixel rate, 1 if line rate, 4 if 1 channel mode a,
//               5 if 1 channel mode b
value = regs[0x26]; // afe op
afeop = value & 7;
if (bShowStatus)
        NSCStatusOut("afeop=%d,%X",afeop,afeop);

value = regs[0x19]; // cis tr timing mode
int tradj;
if (LM9831())
{
        tradj = value & 0x7F;
        value = regs[0xb]/8;
}
ctmode = value & 3;
if (bShowStatus)
        NSCStatusOut("ctmode=%d,%X",ctmode,ctmode);

value = regs[0x51]; // quarter speed count
if (LM9831()) value /=4;
qtcnt = (value & 0xC)/4;
if (bShowStatus)
        NSCStatusOut("qtcnt=%d,%X",qtcnt,qtcnt);

hfcnt = (value & 0x30)/16; // half speed count
if (bShowStatus)
        NSCStatusOut("hfcnt=%d,%X",hfcnt,hfcnt);

value = regs[0x50]; // steps to reverse
strev = value & 0x3F;
if (bShowStatus)
        NSCStatusOut("strev=%d,%X",strev,strev);

value = regs[0x47]; // scan step size lsb
st = value;
value = regs[0x46]; // scan step size msb
st += 256 * (int) value;
if (bShowStatus)
        NSCStatusOut("st=%d,%X",st,st);

//new
BOOL ch3_pix = (afeop == 0);
int en_tradj = 0;
if (tradj) en_tradj = 1;

// calculate dpd
cm = 1;
if (afeop == 1 || afeop == 5) cm = 3; // if 3 channel line or 1 channel mode b
if (bShowStatus)
        NSCStatusOut("cm=%d,%X",cm,cm);

afelat = 7;
if (afeop == 0) afelat = 4;
if (bShowStatus)
        NSCStatusOut("afeop=%d,%X",afeop,afeop);

if (tpsel == 0) tp = 1;
else
{
        tp = tpspd +2;
```

```
                if (tp==5) tp++;
        }
        if (bShowStatus)
                NSCStatusOut("tp=%d,%X",tp,tp);

        b = 1;
        if (ctmode == 0)
        {
                b = (ntr+1)*((2*gbnd)+dur+1);
                if (LM9831()) b+= (1-ntr)*en_tradj;
        }
        if (LM9831()) if (ctmode == 2) b = 3;
        if (bShowStatus)
                NSCStatusOut("b=%d,%X",b,b);

        a = 0;
        if (LM9830())
        {
                if (vps != 0) {
                        a = (linend - actpst - afelat)/vps;
                        a *= tcdp;
                }
        }
        a = linend + a;
        if (bShowStatus)
                NSCStatusOut("a=%d,%X",a,a);

        *tr = cm * (a + tp*(b + 3 - ntr));
        if (bShowStatus)
                NSCStatusOut("tr=%d,%X",tr,tr);

        if (LM9831())
        {
                //new
                int TRadj = tradj;
                if (tradj == 0)
                {
                        if (ctmode == 0) (*tr) += cm;
                }
                else
                {
                        int le_phi,num_byteclk,num_mclkf,tr_fast_pix,extra_pix;
                        if (!ch3_pix)
                        {
                                le_phi = (TRadj+1)/2 + 1 + 6;
                                num_byteclk = ((le_phi + 8*a + 8*b + 4)/(8*TRadj)) + 1;
                                num_mclkf = 8 * TRadj * num_byteclk;
                                tr_fast_pix = num_byteclk;
                                extra_pix = (num_mclkf - le_phi) % 8;
                        }
                        else
                        {
                                le_phi = (TRadj+1)/2 + 1 + 10 + 12;
                                num_byteclk = ((le_phi + 3*8*a + 3*8*b + 3*4)/(3*8*TRadj)) + 1;
                                num_mclkf = 3*8 * TRadj * num_byteclk;
                                tr_fast_pix = num_byteclk;
                                extra_pix = (num_mclkf - le_phi) % (3*8);
                        }
                        *tr = b + a + 4 + tr_fast_pix;
                        if (extra_pix == 0) (*tr)+=1;
                        *tr *= cm;
                }
        }

        if (*tr == 0) dpd = 0;
        else {
                dpd = (((qtcnt*4) + (hfcnt*2) + strev)*4*st) % *tr;
                dpd = *tr - dpd;
        }
        if (bShowStatus)
                NSCStatusOut("dpd=%d,%X",dpd,dpd);

        if (!bWantCM && cm != 0) *tr /= cm; // remove cm component if requested
        return dpd;
}
```

7. Support

For support relating to National Semiconductor scanner products please contact any one of the following individuals:

Jim Brinkhurst - Primary Hardware Support	408 721 4660	James.H.Brinkhurst@nsc.com
Fred Hamilton - Hardware Support	408 721 6010	Fred.Hamilton@nsc.com
Wayne Poole - Hardware Support	408 721 4614	Wayne.Poole@nsc.com
Brian Burford - Primary Software Support	408 721 5931	Brian.Burford@nsc.com
Yani Buchori - Software Support	408 721 2833	Yani.Buchori@nsc.com
Roger Chen - Primary Hardware Support - Asia	886 2 25370261	Roger.Chen@nsc.com
Joseph Hsiao - Primary Software Support - Asia		Joseph.Hsiao@nsc.com