LM9832/LM9833 Software Designers Guide

Revision 2.0

September 2001

Note: Development of this Software Designers Guide was discontinued in 2001. At that time, most of our scanner customers were using our evaluation software as a basis for further software development, rather than starting from scratch using this guide. Please feel free to use the guide as a reference to the operation of the LM9832 and LM9833. It does provide additional information not found in the datasheet or software documentation, but it is not comprehensive, nor is it guaranteed to be 100% accurate. If errors are found, feel free to notify us at scanner.team@nsc.com, and we will address them as soon as possible.

1. Introduction

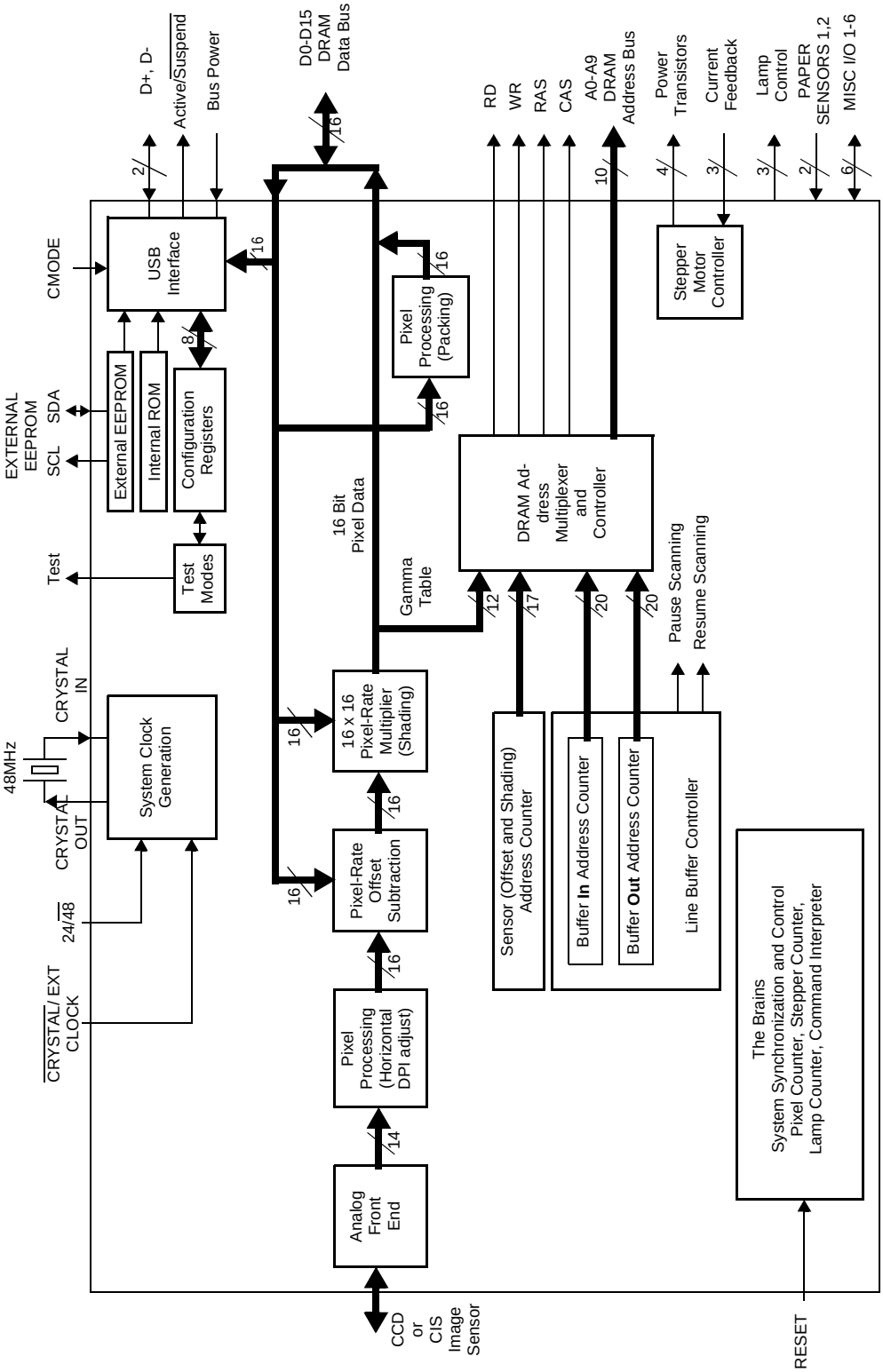
The purpose of this document is to ease the design task of the scanner designer. The LM9832 and LM9833 scanner IC's are complex and powerful. They enable significant advances in system performance with reductions in electrical components. A good first step to utilizing the LM9832/33 devices is to review the datasheet and become familiar with the analog and digital functional blocks of the scanner on a chip product.

1.1 Porting designs from LM9831

Users of the LM9832 or LM9833 may have previous product designs based on the LM9831. The first step in understanding the differences between the previous product and these being discussed is to review the product datasheets. Sections 12 and 13 of the LM9832 and LM9833 datasheets summarize these differences.

2. LM9832/LM9833 Hardware Overview

Figure 1: LM9832 Functional Blocks



2.1 LM9832/LM9833 Features Summary

The LM9832 and LM9833 are highly integrated 14 and 16 bit document scanner ICs with USB interface. The only peripheral ICs necessary to build a complete USB scanner are a DRAM and low cost drive transistors for the stepper motor. In low volume applications, an external serial EEPROM will be used to provide customer specific USB descriptor information.

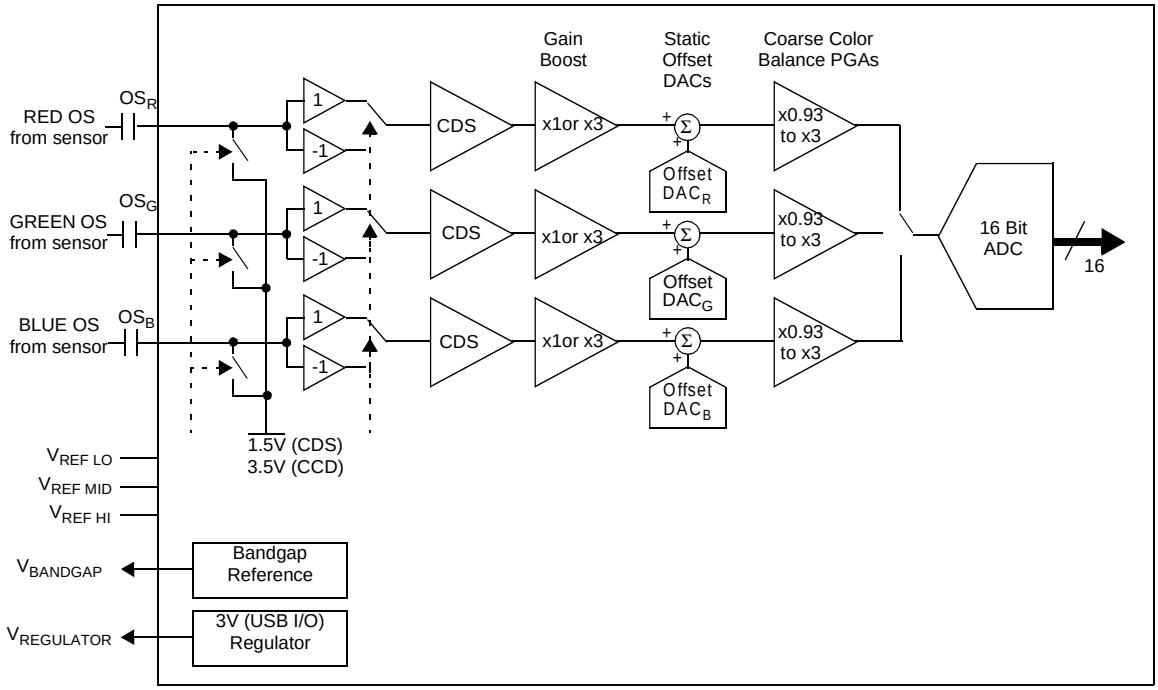
The following major functions are included:

- 14 or 16 bit analog to digital converter operating at up to 6 MSamples/s (2M RGB pixels/s)
- Configurable 3 input analog front end supporting most CCD and CIS sensors
- Fully compliant USB 1.1 Full Speed (12 Mb/s) interface - supports self and bus powered products
- Flexible microstepping motor control system using low cost external drive transistors
- Configurable illumination control for CCD and CIS scanners
- Analog offset/gain correction to maximize system signal to noise performance
- Pixel rate digital offset/gain correction to compensate for sensor and system errors
- Independent 12 bit R, G, B user programmable gamma correction tables
- Configurable pixel depths of 1, 2, 4, 8, and 14 or 16 bits. Lower pixel depths are packed into bytes for faster transmission of lower pixel depth scans
- Register configurable state machine architecture for optimum performance/cost
- Power on Reset default settings for key registers
- Space conserving 100 pin TQFP package

The register based architecture provides a very flexible solution that is completely configurable by the host software. No firmware changes are required for specific applications. Please refer to “Parameter Calculation” on page 32 for more information on the various registers and how to determine specific settings for your application.

2.2 Functional Blocks

2.2.1 AFE



The analog front end performs the following functions.

- Sampling of input waveform from sensor
- Signal polarity adjustment to support CCD and CIS sensors
- Correlated double sampling
- Offset adjustment
- Gain boost and coarse gain balancing
- Analog to digital conversion

2.2.1.1 Sampler, Polarity Selection, Correlated Double Sampling (Registers 0B and 26)

The first section of the AFE captures the analog sensor output signal (s). Three inputs are available, and can be configured to support most CCD and CIS image sensors. Various 1 channel and 3 channel color and monochrome scanning modes are supported. In addition, the polarity of the input signal can be inverted in the sampler to ensure the ADC output always increase with increasing light intensity. Correlated Double Sample (CDS) of signals can be performed to reject common mode signal fluctuations. The operation of the AFE sampling system is controlled via the Color Mode and Sensor Configuration settings.

2.2.1.2 Offset (Registers 38 to 3A)

DC offsets can be applied to each of the AFE input channels. These offsets are adjusted during coarse calibration to optimize the input signal levels into the ADC. A 5 bit plus sign correction value is available for each of the three channels.

2.2.1.3 Gain (Registers 3B to 3D)

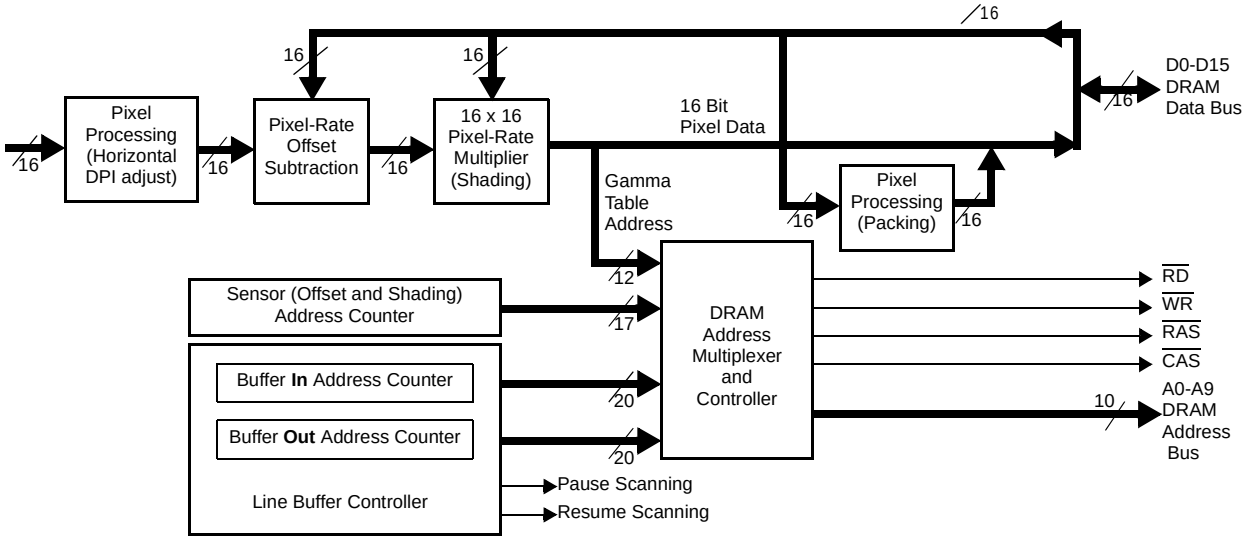
As seen in the block diagram, DC gain is applied at two points in each AFE channel. A x1 or x3 boost gain setting is available, in addition to a 5 bit incremental setting that adjusts gain from 0.93 V/V to 3.0 V/V.

2.2.1.4 ADC

The analog to digital converter always provides a full 16 bit (or 14 bit) result. Further digital pixel processing is performed to generate the desired pixel depth for the scanning application. See “Digital Pixel Processing” on page 6.

2.2.2 Digital Pixel Processing

Digital Pixel Processing is used to perform a variety of modifications to the raw ADC data. The different processes and the reasons for them are discusses in the following sections.



2.2.2.1 Horizontal Dot Per Inch Reduction (HDPI)

During the scanning operation, every pixel is converted by the ADC (except when Preview or Turbo modes are in use). To achieve lower resolution scans, the full resolution data needs to be reduced to that desired by the user. The Horizontal Dot Per Inch reduction block provides reductions of; /1, /1.5, /2, /3, /4, /6, /8, and /12.

Previously, many low cost scanners would simply discard extra pixels, resulting in significant image artifacts when scanning dithered print media. In contrast, the LM983x family of ICs use a digital pixel averaging function to provide even weighting of pixel data, and high image quality in all applications.

The number of pixels coming out of the HDPI reduction block is as shown by this equation:

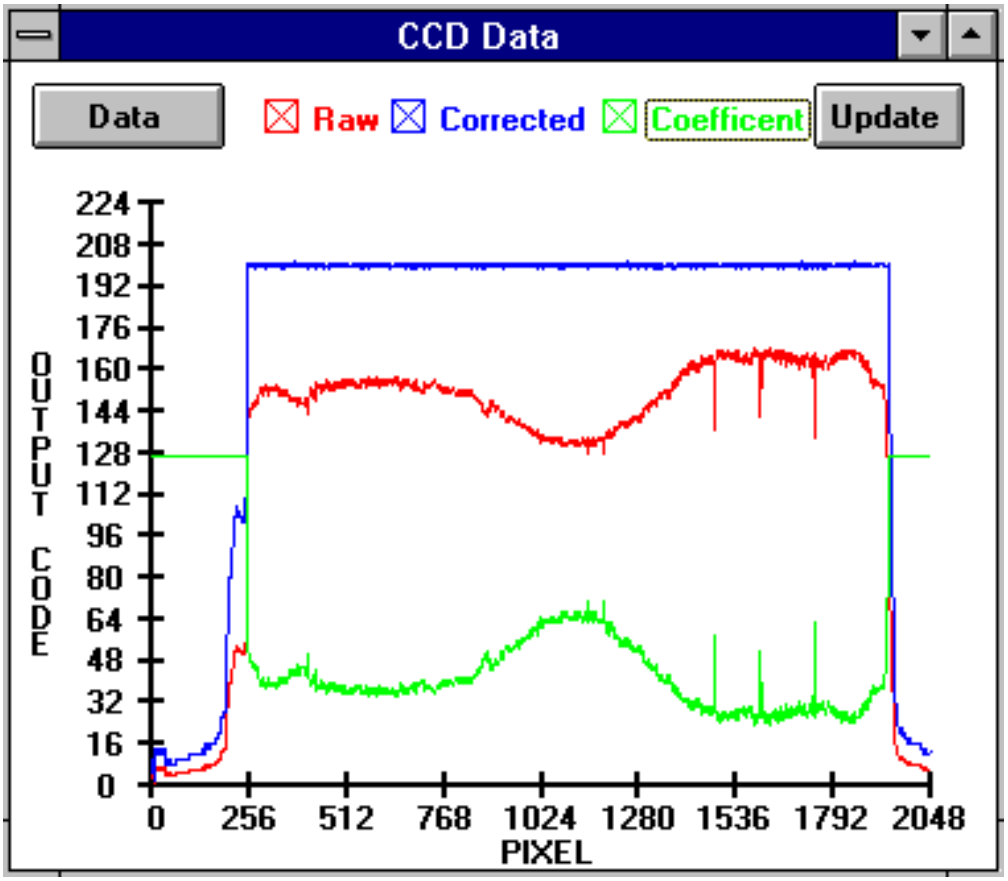
PixelsOut = Int(PixelsIn/DividebyRatio)

If there are not enough pixels at the end of a scan line to form a complete pixel, the last pixel will be eliminated. For example, if a line is 35 pixels wide, and the HDPI reduction setting is /6, then the output of the block will be 5 pixels (The integer portion of 35/6). The last 5 pixels will be discarded, since 6 pixels would be required to form a new pixel in this mode.

2.2.2.2 Pixel-Rate Offset Subtraction/Multiplier

Many error sources exist in the scanner imaging system. Sensors and optics add significant errors that are consistent for every scan line. These kinds of errors can be quantified in a scanning procedure, and corrections can be applied during the scans. The pixel rate offset subtraction and gain multiplier are used to apply the corrections to the RAW scan data. Every pixel will have a unique offset and gain correction coefficient stored in RAM. As each line of data is scanned, the coefficients are retrieved from RAM and applied in the subtraction and gain blocks

Figure 2: Pixel Rate Gain Correction - Before and After (Offset correction already applied).



The offset correction equation is:

$Pixel_{OUT} = Pixel_{IN}$ - Offset Coefficient

The output of the offset correction stage will be between 0 and 65535. Negative outputs are not allowed and will be ‘clamped’ at 0.

The gain correction equation is:

$Pixel_{OUT} = Pixel_{IN} \bullet$ (Offset Coefficient / 16384)

The output of the multiplier can range from 0 to 16383. If the resulting output of the multiplication would have been higher than 16383, the output ‘saturates’ at a value of 16383.

Note that a coefficient of 0 represents a gain of 0, and a coefficient of 16384 represents a gain of 1. The maximum gain is 65535/16384 which is approximately equal to 4.

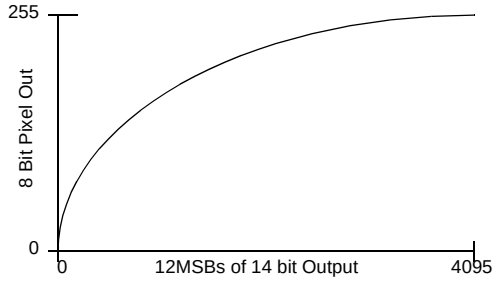
The offset and gain correction tables are loaded into DRAM via the DataPort. See section 6.1 of the datasheet “The DataPort: Reading and Writing to Gamma, Offset, and Gain Memory). The DataPort control register selects which color and type of table will be read from or written to.

2.2.2.3 Gamma Correction Table

There are separate gamma correction tables for Red, Green and Blue data. The input to each table is the 12 most significant bits of the output from the gain correction block. The output of each table is an 8 bit value. The tables consume 12K words (4K per color x 3 colors x 16 bits) of DRAM. Only the 8 MSBs of each word are used as the gamma table output. The tables are useful for several functions. In most

imaging applications, 24 bit (8 bits per color times 3 colors) data is used. One way to achieve aan 8 bit output would be to only use the 8 MSBs of data. However, it is often useful to have a non-linear transfer function to emphasize the data at specific intensities. For instance, it can be useful to expand the dynamic range of the darker image data. This can easily be achieved by using a Gamma correction table as shown in “Example Gamma Correction Curve - Emphasizing Low Intensity Image Data.” on page 8.

Figure 3: Example Gamma Correction Curve - Emphasizing Low Intensity Image Data.



Other gamma curves can be used when generating reduced bit depth data. (Refer to “Pixel Packing (Bit Depth Reduction)” on page 8) When the output data will only be 1 2 or 4 bits per pixel, the Gamma table can be designed to correctly map the 12 input bits to the 1, 2, or 4 output bits. In particular, the transition thresholds of the output data can be controlled to achieve specific grayscale or black/white cutoff points.

Example Gamma Table for 1 bit output and 50% threshold. Note that in 1 bit per pixel mode, the most significant bit of the output of gamma correction is used.

000000000000 => 00000000 (Output pixels are black from 000 to 0x7FF)

000000000001 => 00000000

-
-

011111111111 => 00000000

100000000000 => 10000000 (Transition to white occurs at 50% intensity 0x800)

100000000001 => 10000000

-
-

111111111110 => 10000000

111111111111 => 10000000

The gamma tables are loaded through the dataport. See section 6.1 of the datasheet “The DataPort: Reading and Writing to Gamma, Offset, and Gain Memory). The DataPort control register selects which color and type of table will be read from or written to.

2.2.2.4 Pixel Packing (Bit Depth Reduction)

The Pixel Packing function is used when the pixel data has a bit depth of less than 8 bits. Some scans require only one bit per pixel (“line art” mode), others may need only 2 or 4 bits/pixel. To increase scanning speed for lower pixel depths, the LM9832 packs the desired MSbs of multiple pixels together into

one 16 bit word, increasing the transmission speed to the host by a factor of 2, 4, 8, or 16. Figure 1 shows how the pixels are packed together for 8, 4, 2, and 1 bit pixel depths. In Figure 1, “b” indicates the bit position (b7 = the most significant and b0 = the least significant bit) of the original 8 bit pixel data, and p_n indicates the original pixel sequence, i.e p_0, p_1, p_2, p_3...

If there are not enough unpacked pixels at the end of a line to complete the packed word for transmission, that final word is not sent. For example, doing an 8 bit pixel rate scan with a HDPI divider of 1 and an odd number of pixels will truncate the blue component of the last pixel.

Table 1: Packing Multiple Pixels Into One Word

	Bit Position and Data Within 16bit Word							
Pixel Depth	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
8	b7 p_0	b6 p_0	b5 p_0	b4 p_0	b3 p_0	b2 p_0	b1 p_0	b0 p_0
4	b7 p_0	b6 p_0	b5 p_0	b4 p_0	b7 p_1	b6 p_1	b5 p_1	b4 p_1
2	b7 p_0	b6 p_0	b7 p_1	b6 p_1	b7 p_2	b6 p_2	b7 p_3	b6 p_3
1	b7 p_0	b7 p_1	b7 p_2	b7 p_3	b7 p_4	b7 p_5	b7 p_6	b7 p_7
	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
8	b7 p_1	b6 p_1	b5 p_1	b4 p_1	b3 p_1	b2 p_1	b1 p_1	b0 p_1
4	b7 p_2	b6 p_2	b5 p_2	b4 p_2	b7 p_3	b6 p_3	b5 p_3	b4 p_3
2	b7 p_4	b6 p_4	b7 p_5	b6 p_5	b7 p_6	b6 p_6	b7 p_7	b6 p_7
1	b7 p_8	b7 p_9	b7 p_{10}	b7 p_{11}	b7 p_{12}	b7 p_{13}	b7 p_{14}	b7 p_{15}

The gamma table in “Pixel Packing (Bit Depth Reduction)” on page 8 allows the user to set the threshold of each transition for various line art or reduced pixel depth modes.

The LM9832 also supports a 14 bit out mode while the LM9833 supports a 16 bit mode. These can be used to get very accurate data for calibration or to scan a 14 (16) gray/42 (48) bit color image. This mode is set through register 9, bit 5. In the 14 (16) bit output mode, the gamma and pixel packing stages are bypassed, and the 14 (16) bit data from the ADC is stored in DRAM, formatted as shown in Figure 2.

Table 2: 14 (16) Bit Output Mode Data Format

MSB	15	14	13	12	11	10	9	8
LM9832	b13	b12	b11	b10	b9	b8	b7	b6
LM9833	b15	b14	b13	b12	b11	b10	b9	b8
LSB	7	6	5	4	3	2	1	0
LM9832	b5	b4	b3	b2	b1	b0	0	0
LM9833	b7	b6	b5	b4	b3	b2	b1	b0

In the LM9832, the 14 bit data is stored as a 16 bit word, with the 2 least significant bits set to 0.

Note: The memory reserved for the gamma table is used to store image data in the 14 bit mode. After scanning in 14 bit mode, the gamma table must be reloaded for operation in 8, 4, 2, or 1 bit mode.

2.2.2.5 Data Buffering

The line buffer uses the external DRAM as a FIFO line buffer to store the pixel data (which is generated at a fixed rate, synchronous to the CCD clocks) and send it back to the PC at an asynchronous, unpredictable, and non-constant rate.

The LM9832 supports 2 sizes of DRAM, 256k x 16bit and 1M x 16bit. 216kbytes (108kwords) of the capacity of the DRAM is consumed by the offset and shading coefficients and the gamma tables. That

leaves 296kbytes of memory available for line buffer when using a 256k x 16 bit DRAM, or 1832kbytes of memory when using a 1M x 16 bit DRAM.

The line buffer is tightly coupled to the stepper motor (“Stepper Motor Controller” on page 17), and is responsible for stopping the motor before the buffer overflows and starting the motor again as the buffer nears empty.

If the scanner is generating pixel data faster than the PC can acquire it, the line buffer will start to fill up. As the buffer nears 100% of its capacity, the scan must be paused before it starts acquiring a line which will overflow the buffer. This Pause Threshold limit (register 4E) is programmable in 2 kbyte (256k x 16 bit DRAM) or 8kbyte (1M x 16 bit DRAM) increments between 0 and 255.

To maximize scanner performance and minimize pausing due to buffer full conditions, the pause threshold should be set using this formula:

Pause Threshold (kB) = Available_Memory - (Line_Length + 1)

where Available_Memory (kB) = 296kbytes (256k x 16b DRAM) or 1832kbytes (1M x 16 bit DRAM),

Line_Length (kB) = (Bytes/Line)/1024

$$\text{Bytes/Line} = 2 \cdot \text{INT}\left(\frac{\text{INT}\left(\frac{\text{Data Pixels}}{\text{HDPI\_Divider}}\right) \cdot C \cdot B}{16}\right)$$

Where C = 1 for “1 Channel Grayscale”, 3 for all other modes,

Data_Pixels = Data Pixels End (registers 24, 25) - Data Pixels Start (registers 22, 23)

HDPI_Divider = Horizontal DPI divider = 1, 1.5, 2, 3, 4, 6, 8, or 12

B = Bits per Pixel = 16 (14 bit mode), 8, 4, 2, or 1

Register 4E value = Pause Threshold (kB)/2 (256k x 16 DRAM) or Pause Threshold (kB)/8 (1M x 16 DRAM)

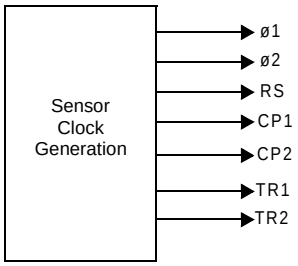
When the Pause Threshold is reached the buffer sends a command to the stepper motor controller to stop scanning. The remainder of the line being processed will continue being processed and be sent to the buffer. If the Lines To Process After Pause Scan Signal register (register 54) is greater than 0, then room for these additional lines needs to be added into the Pause Threshold value calculation.

Note that the scanner software on the host PC must set a Pause Threshold value low enough to ensure that any data that comes after a pause request (the rest of the current line and any subsequent lines if register 54 bits 0-2 are greater than 0) will fit into the DRAM buffer. If the Pause Threshold is set too high, the Line Buffer may overflow, creating discontinuities in the scanned image.

After a pause, the buffer will continue to transmit data to the PC until it hits the Resume Threshold limit (register 4F), which is also programmable in 2 kbyte (256k x 16 bit DRAM) or 8kbyte (1M x 16 bit DRAM) increments between 0 and 255. When the Resume Threshold is reached, the Line Buffer sends the motor controller a command to resume.

2.2.3 Sensor Timing Generation

Figure 4: Sensor Control Signals



This function generates the clock signals necessary to control a CCD or CIS sensor. Seven signal outputs are available for driving the various CCD and CIS timing control lines. The primary names of these signals are as shown in Figure 4. Depending on the various register settings, these outputs may have alternative functions to the primary names shown.

There are three different types of settings that control the operation of the sensor control signals.

- Mode controls - Controls that affect the overall operation of the sensor control signals. (Registers 0B, 0D, 1C-25, 26, 27)
- Line rate controls - Controls that affect the timing and operation of signals that happen once per Line or once per TR pulse. (Registers 0B, 0D, 1C-25, 26, 27)
- Pixel rate controls - Controls that affect the timing and operation of signals that occur once per pixel. (Registers 0B, 0C, 0D, 0F-18, 26)

To prevent sensor saturation, the LM9832 is always clocking the CCD/CIS, except when it is in Reset or Standby (Register 7 bit 2 or 3 = 1).

Refer to the descriptions for registers 0B to 27 for more details on the timing of specific signals. Also note that the AFE sample timing is synchronous to the sensor control signals and is controlled by many of the same register settings. Figure 5 shows timing signals generated for typical CCD and CIS sensors.

Figure 5: Typical Line Rate Timing of CCD and CIS Control Signals

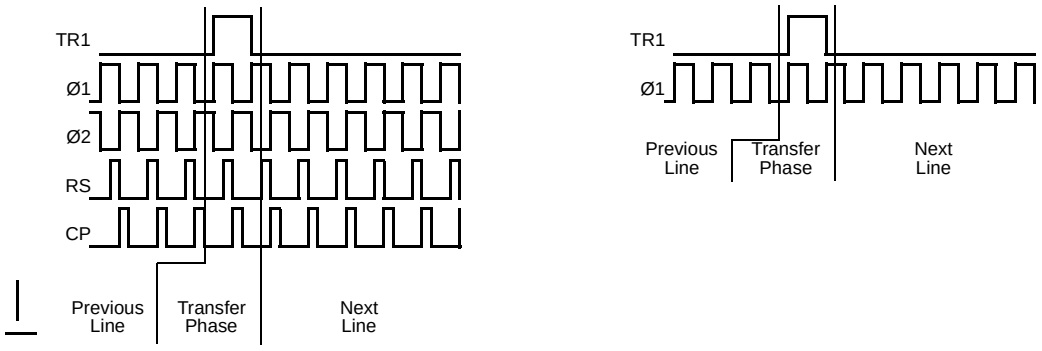
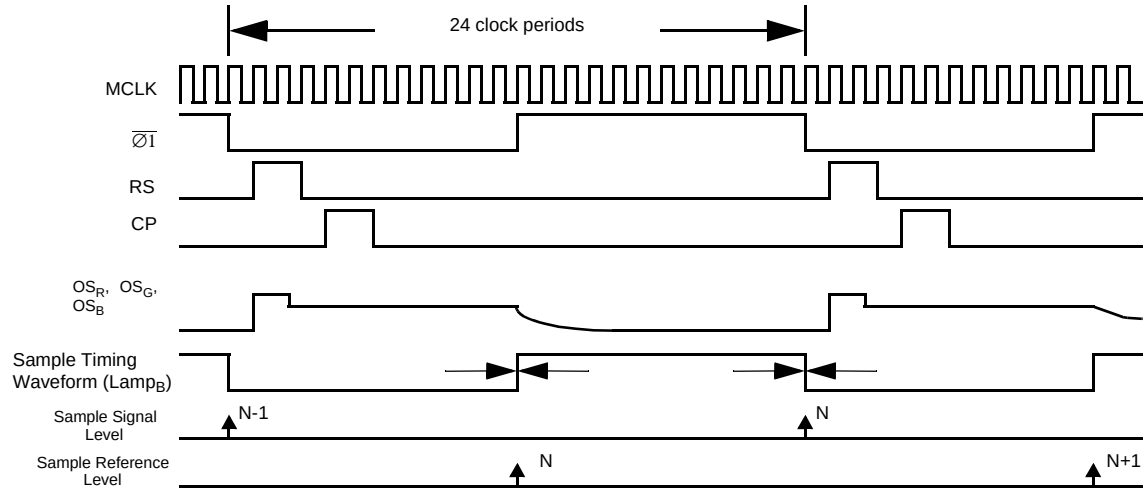
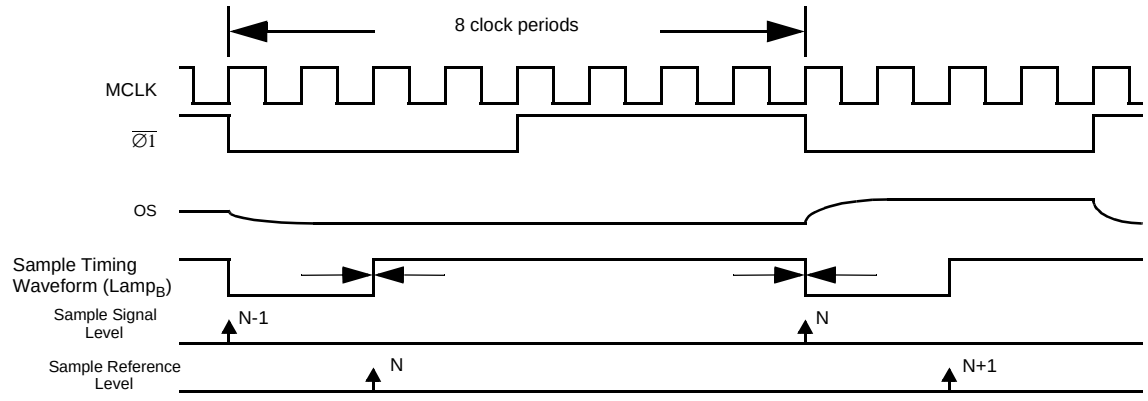


Figure 6: Typical Pixel Rate Timing of CCD and CIS Control Signals



Typical CCD Application



Typical CIS Application

The detailed pixel rate timing of the sensor control signals is adjustable in increments of one MCLK via registers 0F through 18. Note that there are 24 MCLKs per pixel period in “Pixel Rate Color” mode (see Register 26 descriptions) which is only used with 3 output CCD sensors. In all “Line Rate Modes”, which can be used with CCD or CIS sensors, there are 8 MCLK cycles per pixel period. When adjusting the timing of the sensor control signals it is important to remember that the adjustment range changes when switching modes.

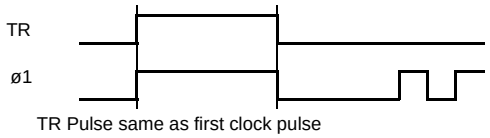
The CDS (correlated double sampling) timing is also adjustable. A test mode provides added visibility of the CDS timing via the $Lamp_B$ output pin. When bit 7 of register 5E is set, the sample timing is seen as a square wave on the $Lamp_B$ output. The positive edge of the square wave is the reference sample point, while the negative edge is the signal sample point. When CDS is off, an internal voltage reference ($V_{REF\ LO}$ or $V_{REF\ HI}$ dependent on the polarity setting in register 0B) is sampled instead of the OS input.

Key features of the sensor control signal block to support specific sensors include:

- Independent control over the polarity (inverting or noninverting) of the input stage to accommodate CIS or CDS signals.

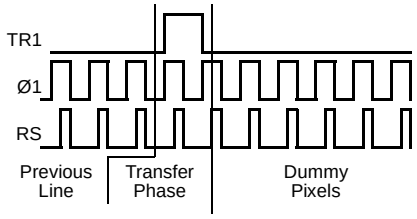
- Full timing control of the CIS and CDS sample points. Reference and signal sample points can be independently adjusted. Note that the absolute time between reference sample and signal sample must be 2 MCLKs or greater, whether CDS is on or off.
- Ability to turn off CDS. When CDS is on, traditional CDS is performed. When CDS is off, the signal is sampled at the Sample Signal point, but an internal voltage reference (either $V_{REF\ HI}$ or $V_{REF\ LO}$, depending on the signal polarity setting) is used for the Sample Reference voltage (not a point on the input signal itself).
- The CP1 output supplies the CP pulse needed on some popular Toshiba CCDs. This looks and acts just like another, independent RS pulse.
- A CP2 output is another independent pixel rate pulse that (if needed) can be programmed to supply an additional clock.
- CCD clock signals RS, CP1, CP2 are reset when Line Ends
- The internal Clamp signal is reset with Optical Black Pixels End.
- TR1 and TR2 pulse widths are always the same width, as determined by Register 0E.
- The TR-Ø1 guardband may be equal to 0, causing TR and Ø1 to go high simultaneously and low simultaneously (Figure 7). This is a requirement of some Canon CIS sensors.

Figure 7: TR-Ø1 Guardband Can Be Equal To 0



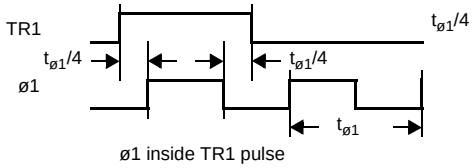
- CIS TR1 Timing Mode 1. In this mode the TR1 pulse is exactly one Ø clock long, occurring on the rising edge of Ø1. The TR1 pulse width and guardband settings are ignored. For Dyna CIS.

Figure 8: CIS TR1 Timing Mode 1



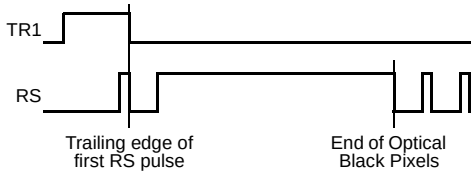
- CIS TR1 Timing Mode 2. In this mode the TR pulse is again equal to 1 Ø period, but now it is centered around Ø1. The TR pulse width and guardband settings are ignored. For Canon CIS.

Figure 9: CIS TR1 Timing Mode 2



- There is a bit for **Fake Optical Black Pixels** (register 19, bit 2). This is used with Dyna CIS sensors. In this mode, the RS output pulses once inside the TR1 pulse, then is held high until the end of the optical black pixels. The TR1 pulse is extended until the trailing edge of the first RS pulse. This mode works for TR1 only, under all TR1 settings (normal and CIS TR1 Timing modes 1 and 2).

Figure 10: Fake Optical Black Pixels



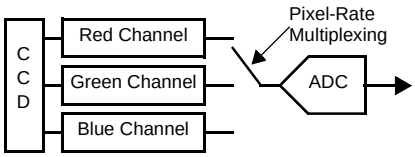
2.2.4 AFE and Sensor Control Signal Timing Modes

Since the AFE and Sensor control signals are tightly interlinked, the various color modes affect the operation of both sections of the IC.

The LM9832 supports the following operation modes, controlled by registers 26 and 27:

- 3 Channel Pixel Rate Mode. In this mode all three channels are converted with the multiplexer in front of the ADC switching at the ADC conversion rate, producing interleaved RGB data that is transferred to RAM. The ADC runs at MCLK/8, each channel's pixel rate is MCLK/24. Each color has its own offset and gain coefficients. This mode typically uses Illumination Mode 1.

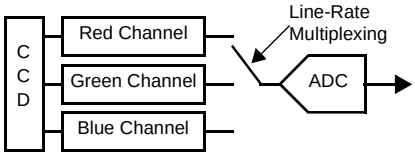
Figure 11: 3 Channel Pixel Rate Mode



ADC Out LIne 1: RGBRGBRGBRGBRGB...
ADC Out LIne 2: RGBRGBRGBRGBRGB...
ADC Out LIne 3: RGBRGBRGBRGBRGB...
ADC Out LIne 4: RGBRGBRGBRGBRGB...

- 3 Channel Line Rate Mode. In this mode all three channels are converted with the multiplexer in front of the ADC switching at the line rate, producing a line of Red data, followed by a line of Green data, followed by a line of Blue data, etc. that is transferred to RAM. The selected channel and the ADC both run at MCLK/8. Each color has its own offset and gain coefficients. This mode typically uses Illumination Mode 1.
- Note: Since the AFE (and therefore pixel rate) are running 3 times faster than for 3 Channel Pixel Rate Mode, this mode can be used to reduce the integration time for high resolution CCD sensors scanning at the full resolution. This reduces integration time while still meeting the data bandwidth requirements of the DRAM interface.

Figure 12: 3 Channel Line Rate Mode

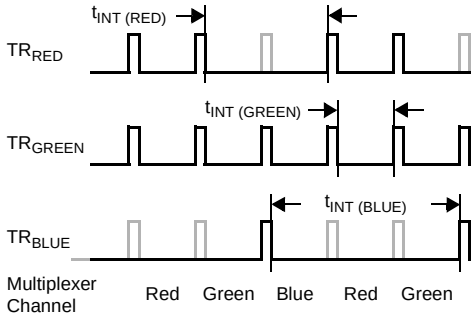


ADC Out LIne 1: RRRRRRRRRRRRRRR...
ADC Out LIne 2: GGGGGGGGGGGGGG...
ADC Out LIne 3: BBBBBBBBBBBBBBBB...
ADC Out LIne 4: RRRRRRRRRRRRRRR...

- In the 3 Channel Line Rate Mode three TR pulses are generated. TR_{RED} is the TR1 output, TR_{GREEN} is the TR2 output, and TR_{BLUE} is the CP2 output. In this mode TR pulses for a particular color can be "skipped", increasing the integration time for that color. In the example shown in Figure 13, the red channel sees 2 times the integration time of the green channel, and the blue channel sees 3 times the inte-

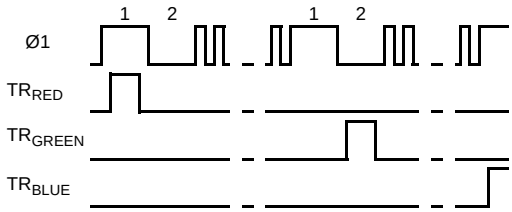
gration time of the green channel. Each channel can be independently programmed to drop 0, 1, or 2 TR pulses.

Figure 13: 3 Channel Line Rate TR Pulse Timing



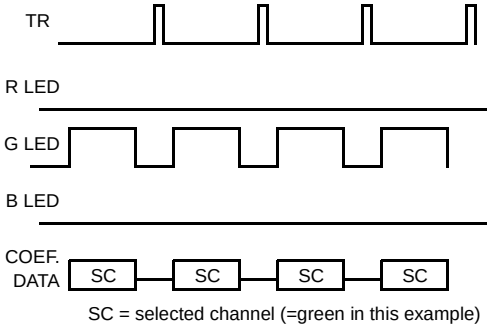
- Each color's TR pulse can be programmed to occur in position 1 (inside Ø1 high) or position 2 (inside Ø1 low), as shown in Figure 14.

Figure 14: 3 Channel Line Rate Mode with 2 TR Pulse Positions



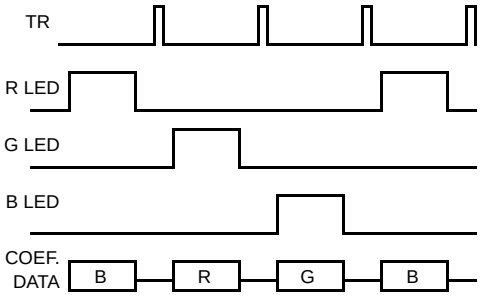
- 1 Channel Grayscale: Uses the selected channel's offset and gain coefficients for all lines. 1 Channel Grayscale is used to scan a grayscale images. This mode typically uses Illumination Mode 1 when used with a 3 Channel Color sensor, or Illumination Mode 3 when used with a 1 Channel sensor.

Figure 15: 1 Channel Grayscale



- 1 Channel Color: This mode uses a sensor tied to the Blue OS input only. Illumination is switched in RGBRGB pattern at the line rate. Each color has its own digital offset and gain coefficients as well as static Gain and Offset data. Note that there is a one line delay between when a line is exposed to a color and when pixels of that color are clocked out of the sensor. For example, the Green LEDs should be on while you are clocking out Red pixels. This mode uses Illumination Mode 2.

Figure 16: 1 Channel Color



2.2.5 Preview and Turbo Modes

These modes actually existed in the LM9831, but were not documented.

The Turbo and Preview modes allow additional pixel averaging (horizontal resolution reduction) to be done in the analog domain. This can be useful, for example, when you have a 1200 dpi scanner and wish to scan at 75 or 50 dpi. The HDPI divider function's lowest resolution is divide-by-12. With the HDPI divider set to divide-by-8 and turbo or preview mode set to x2, the horizontal resolution will be 1200/16 = 75 dpi. With the HDPI divider set to divide-by-12 and turbo or preview mode set to x2, the horizontal resolution will be 1200/24 = 50 dpi. The HDPI divider and Turbo/Preview modes can be used in any combination.

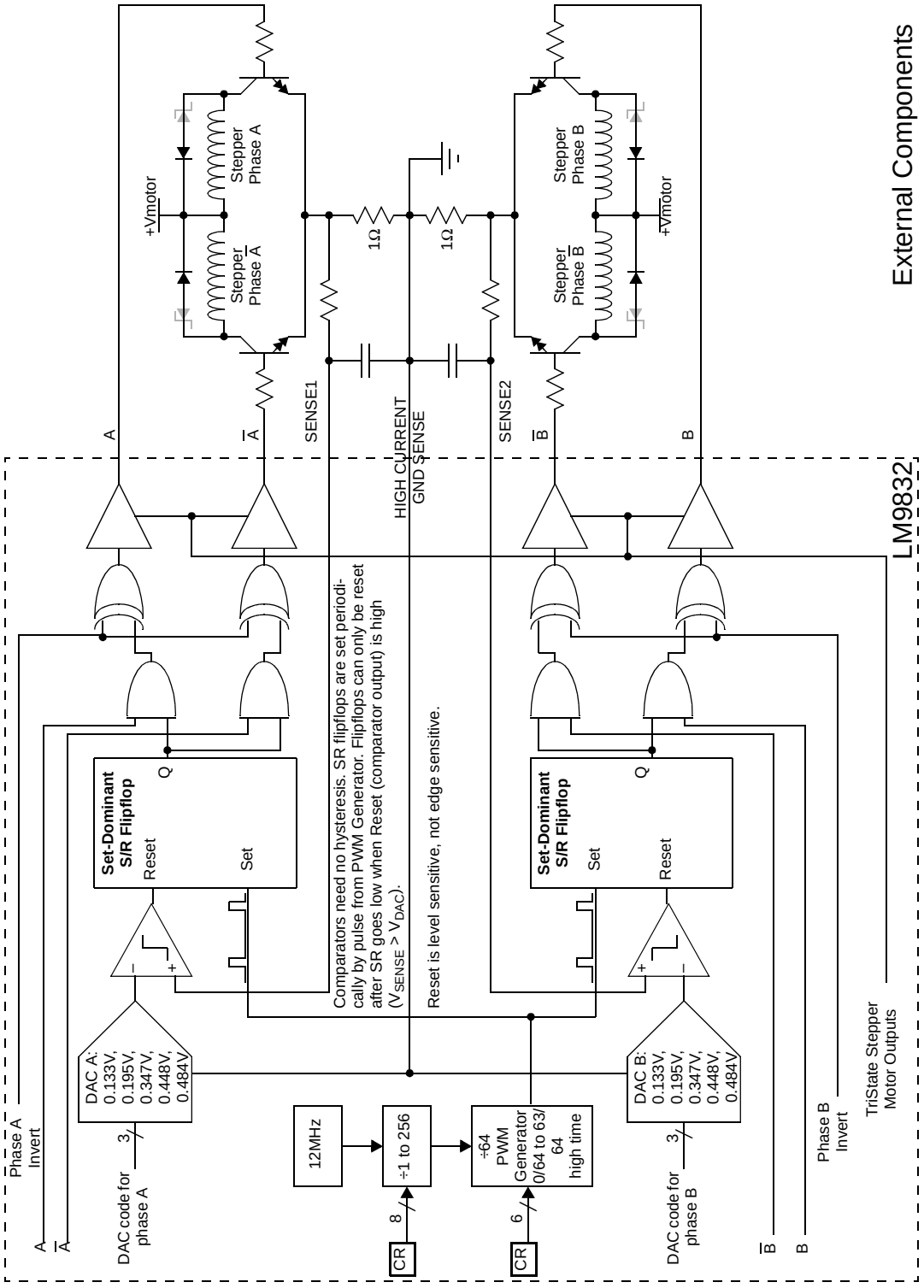
For a Preview factor of xN, the Preview Mode operates by increasing the pixel clocks to the CCD by a factor of N, while suppressing (N-1) reset pulses out of every N pixels. This is only useful for CCDs (or CIS sensors made with CCD technology).

In the Turbo Mode, the entire analog front end is run N times faster, and every N pixels are averaged together before they are converted to digital by the ADC. When using Turbo Mode, the range of registers 0F to 18 is reduced by the Turbo Mode factor, according to the following table.

Table 3: Turbo Mode Timing Limitations

Mode	Pixel Rate	Registers 0F to 18 Range
3 Channel, Turbo off	MCLK/24	0-23
3 Channel, Turbo x2	MCLK/12	0-11
3 Channel, Turbo x3	MCLK/8	0-7
3 Channel, Turbo x4	MCLK/6	0-5
3 Channel, Turbo x6	MCLK/4	0-3
1 Channel, Turbo off	MCLK/8	0-7
1 Channel, Turbo x2	MCLK/4	0-3

2.2.6 Stepper Motor Controller



- Stepper Motor (include notes re: pull-downs on outputs for FET type drivers, plus information related to settings of PWM frequency and minimum duty cycle)

- Note: Acceleration and Deceleration. When decelerating from Fast Forward to Stop or to Scan the deceleration profile uses the Scan stepsize instead of the Fast Feed stepsize. This can cause the deceleration to take significantly longer than expected when scanning at high resolutions.

2.2.7 Lamp Controller

The lamp controller block provides several different modes to support the different types of lamps that will be used. CCD sensors typically use a CCFL (Cold Cathode Fluorescent Lamp) for direct, or transmissive illumination, while CIS sensors usually have an integrated 3 color LED assembly for reflective illumination only.

Register 29 is used to select the mode for the Lamp Control outputs.

- Mode 0 is the Off default reset state.
- Mode 1 is the CCFL mode and provides a duty cycle controlled PWM output on the LampG output pin. The duty cycle is set using registers 2A and 2B. This can be used to control a variable intensity drive for the CCFL lamp.
- Mode 2 is the CIS mode and provides a line rate duty cycle control for operating LED lamps. The On and Off points (in units of pixels) for the Lamp outputs are set using registers 2C through 37. The Red, Green and Blue outputs will alternate at the line rate. This can also be used to control the electronic shutter feature on certain CCD type sensors.
- Mode 3 is similar to Mode 2. The On and Off points (in units of pixels) for the Lamp outputs are set using registers 2C through 37. The Red, Green and Blue outputs will are all on for each line. Care should be exercised using this mode as many CIS sensors are only designed to have one LED on at a time. This can also be used to control the electronic shutter feature on certain CCD type sensors.

2.2.8 Paper Sense and Misc. I/O

Two dedicated Paper Sense inputs are provided to control scanner movement. These can be used for the home position sensor, to provide repeatable system movements when scanning or fast feeding. They are also useful when terminating scans in sheetfed scanner systems. Register 58 is used to configure the basic functions of the Paper Sense inputs. Registers 4C and 4D provide additional movement options in conjunction with bit 5 of Register 58. These enable programmed distance movements via scan or fastfeed commands or upon the triggering of the Paper Sense 2 input.

Six miscellaneous Input/Output pins are provided for general purpose functions within the scanner product. All six pins can be configured as inputs or outputs, however Misc. I/O 1,2 and 3 have power on defaults input settings, while Misc. I/O 4,5 and 6 default at outputs. There are further differences in the power on default settings that provide the user with flexibility in default settings for output functions like LEDs.

The Paper Sense and Misc. I/O pins are also part of the LM9832/3 USB interrupt system. The status of these pins can be accessed by reading Register 2. Changes in the state of these inputs can be configured to cause a USB interrupt, which can then trigger scanning processes in the host system. This feature can also be used to initiate a USB Wakeup from the USB Suspend state. Please refer to the LM9832/3 USB driver software documentation for more details on this feature.

2.2.9 Test Modes and Diagnostic Tuning Features

Registers 5C, 5D, and 5E are used for diagnostic and testing of the ADC and pixel processing sections of the IC. Registers 5C and 5D are used to provide a fixed input to the HDPI divider for debugging pixel processing and software post processing of data. Register 5E selects the following:

- operational mode of the AFE and ADC
- source for data input to the HDPI processor
- use of 16 bit counter as input to HDPI processor
- mode of 16 bit counter
- MCLK edge for AFE (always set to 0)
- CDS sample timing available on LampB output pin

The AFE/ADC operational mode selection is used during device characterization and testing.

The various options related to HDPI source data are useful for device testing, and for debugging of pixel data post processing software.

The CDS sample timing can be output on the LampB output pin. This is very useful when setting the CDS sample timing in Registers 17 and 18. The CDS sample timing can be viewed on an oscilloscope in direct relation to the OS signal inputs to the AFE. This allows the best sample timing to be selected very easily without having to do extensive data evaluation and repetitive tests.

2.2.10 Version Number

The three LSBs of Register 69 indicate which LM983X device this is. This information may be useful to allow software to differentiate between the LM9831 and the LM9832/3 devices.

Table 4: LM983X Version Numbers

Register 69 Value	Device
010	LM9830
011	LM9831
100	LM9832 or LM9833

2.2.11 Power-on-Reset Default Register Settings

The LM9832/3 control registers occupy 128 single byte locations from 00h to 7Fh. Key registers within this space are assigned power on default settings. The power on default settings are designated by the highlighted boxes in the datasheet register descriptions. Once initial USB communications has been established, and before setting the command register to anything other than the Reset state, it is important to load the ENTIRE register space with the appropriate values, including the proper values for all reserved and test registers. This must be done to ensure proper operation of all state machines.

2.2.12 Procedure for Adjusting Register Settings

The LM9832 and LM9833 devices contain a number of different state machines, some of which operate asynchronously to one another. To ensure proper operation, some register settings can only be adjusted when the chip has been placed in the Idle, or Reset states. Please refer to the following table for details.

Table 5: Register Write Rules

Command Register Setting (Reg 07)	Registers	Notes
Any	07	Command Register Only

Command Register Setting (Reg 07)	Registers	Notes
Idle (0x00)	03-06, 2A-37, 38-3D, 42, 45, 58-5B	
Reset (0x20)	All others exept Read Only below	See procedure below.
None	0, 1, 2	Read Only Registers

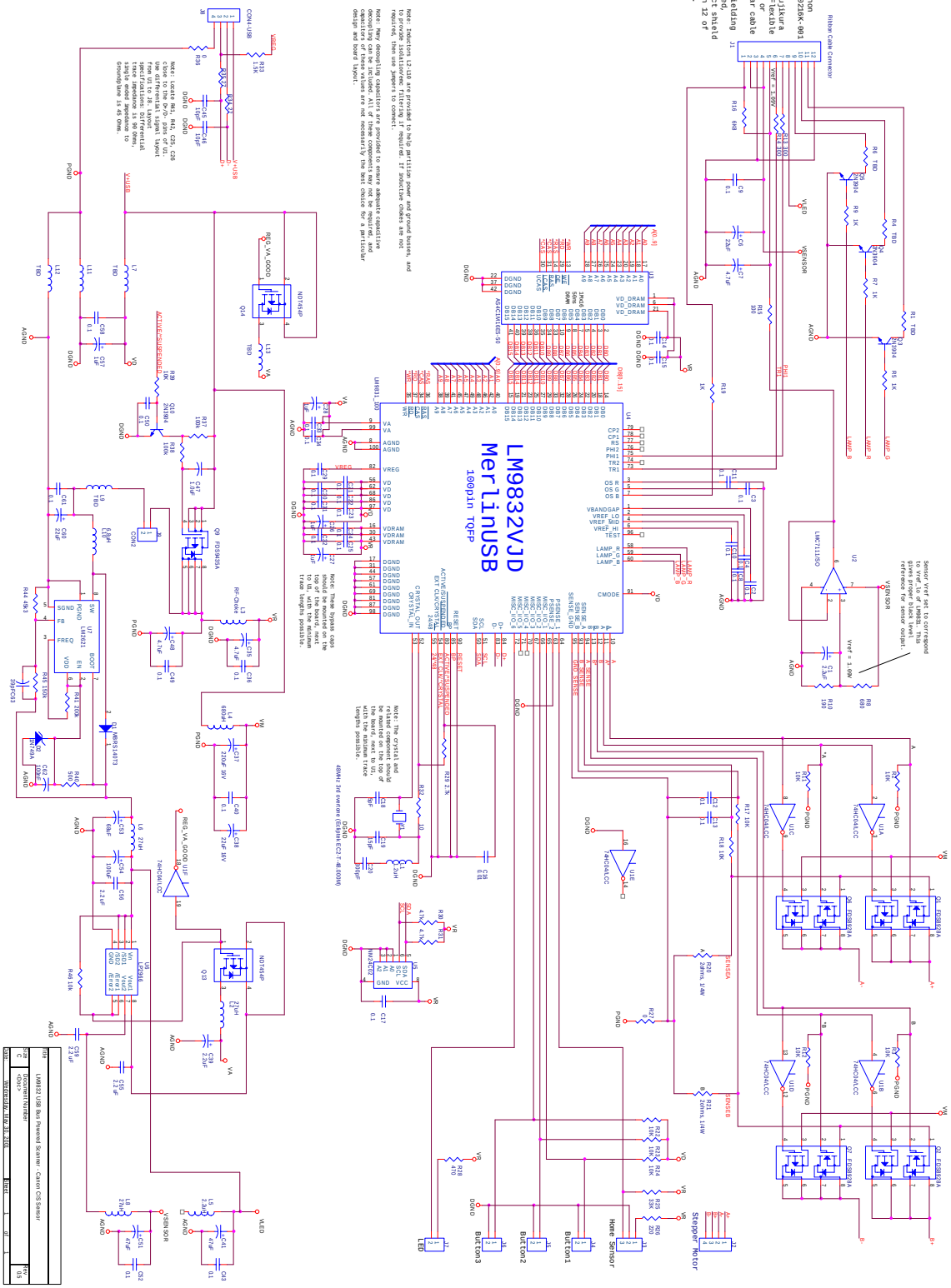
The following procedure should be followed when entering the Reset (Bit 5 of Reg 07 = 1) state.

- Set Register 07 to 0x00 (Idle)
- Set Register 18 to 0x18 (Disable AFE sampling)
- Delay (Minimum delay of 32.5 us, this allows the AFE to finish sampling the current pixel)
- Set Register 07 to 0x20 (Reset)
- Write original value back to Register 18.
- Write other registers as required.
- Set Register 07 to 0x00 (Idle)

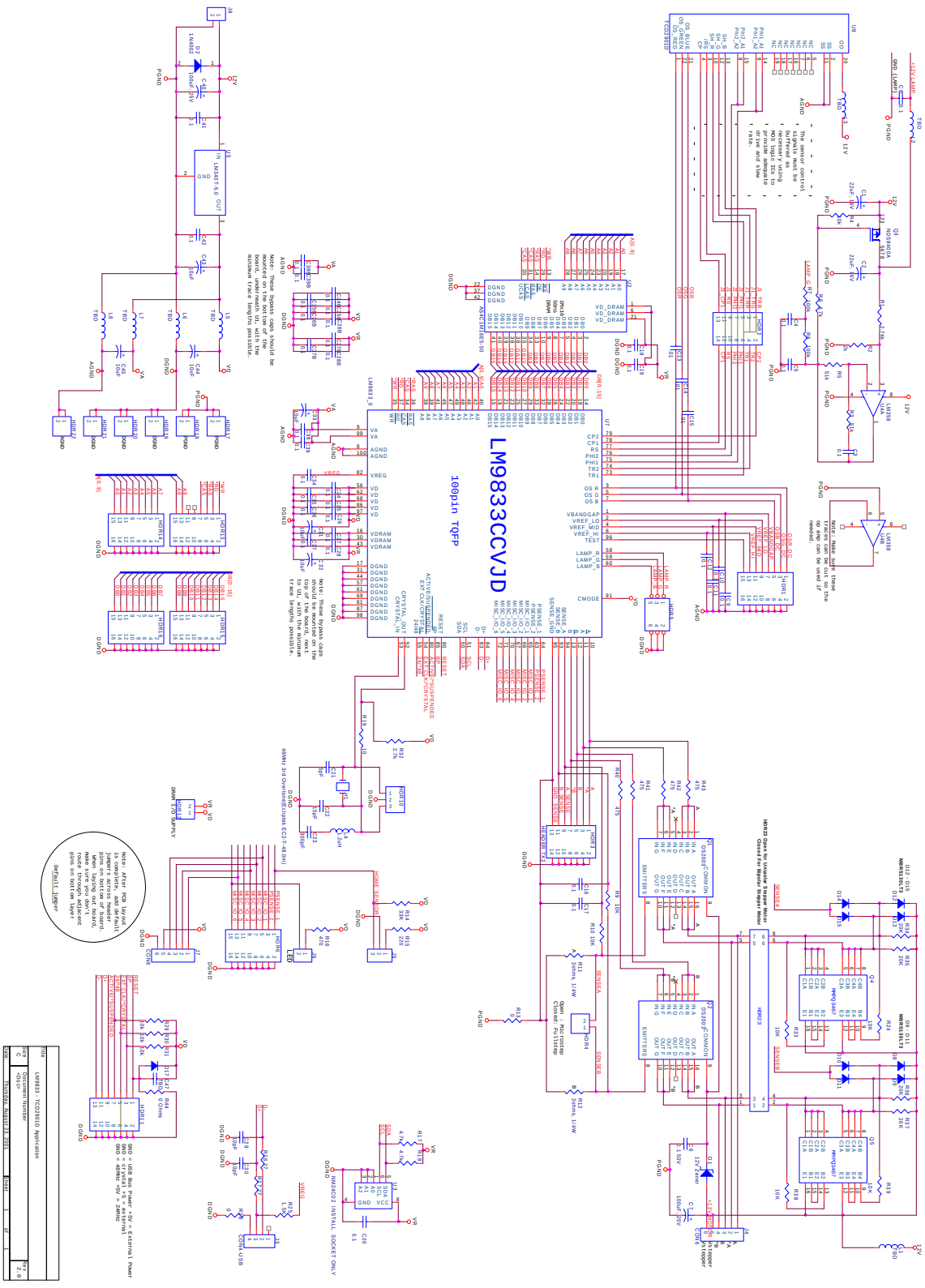
Any new other command can now be issued from the Idle state. The above procedure should always be following when entering the soft Reset state.

3. Example System Schematics

3.1 CIS Based Bus Powered System



3.2 CCD Based Self Powered System



4. Scanner Operation

4.1 Power Applied (Self Powered)

A self powered USB device is NOT powered via. the USB cable. Power is usually supplied by an external DC power supply, and is often further regulated within the scanner. Typically a 12V DC voltage is supplied and regulated to 5V DC to supply the analog and digital components inside the scanner. The 12V is used to power the CCD sensor and stepper motor circuitry.

The +5V analog and digital supplies for the LM9832/3 should be supplied from a common voltage regulator source and have independent ground/power planes and de-coupling capacitors. See the datasheet for more information on this topic.

When power is applied, the scanner IC performs an internal power-on sequence. During this operation, a limited number of key registers are initialized to default values. The power on default values of these registers ensure that the chip is in a safe state after power up. The stepper motor, Misc. I/O and other outputs are in known states. Because most registers do not have default settings, it is important to load ALL registers (0x01h to 0x7Fh) with known values before the command register is taken out of the reset state.

After the power on sequence, the IC waits for communication on the USB interface. If the USB interface is inactive for more than 3.0 ms (i.e. if no cable is connected and the recommended pullup resistor on D+ is installed) then the device will transition from the active to the suspended state.

When activity is sensed on the USB data bus, the IC will return to the active state in accordance with the USB 1.1 specification.

4.2 Connect (Self Powered)

When a connection is detected on the upstream port, the scanner IC returns to the active state. It is then reset by the host, is assigned a unique device address and is configured by the host. Once the device has been configured, it is ready to be accessed by the application software.

4.3 Connect (Bus Powered)

In a bus powered application, the connection event is coincident with the application of power to the device. In these applications, the system must conform to the USB power specifications found in Section 7.2 of the USB specification which is available at www.usb.org. There are a number of different types of specifications that apply. These include, DC power consumption for each of the different device states, Inrush current, and power control during connect/disconnect and suspend events.

4.4 Scanning and Calibration

The following flowchart illustrates 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 17: Basic Scan Flowchart (for CCD Sensors)

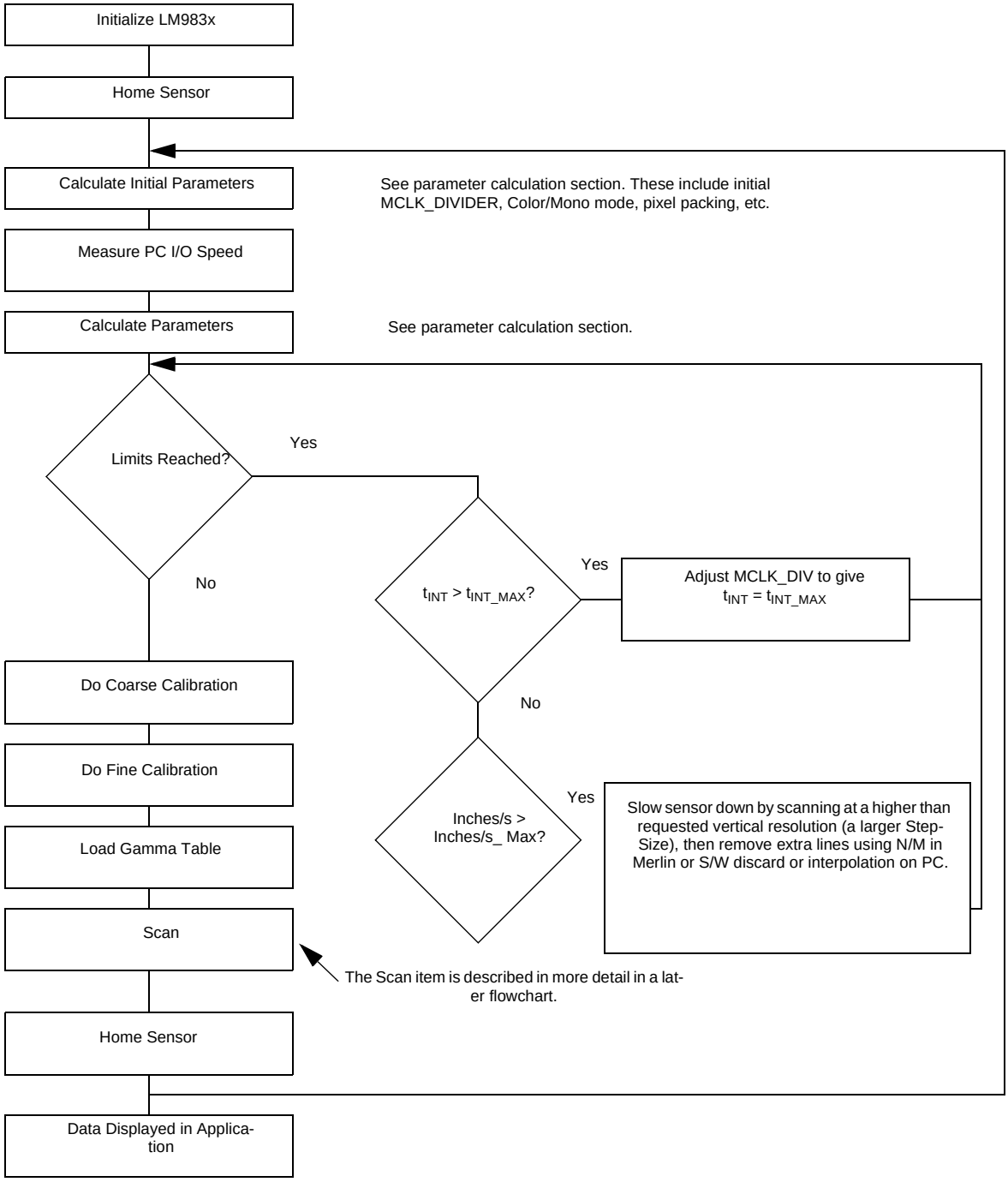


Table 6: Function Descriptions - Major Operations and Related Software

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

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. LM9832 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 grayscale 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 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. Alternatively, black and white full width strips can be used to provide black and white calibration source images for all pixels. [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

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 LM9832). 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 grayscale 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 area. [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 area. 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 Scan-StateMachine.cpp DoCalibration() in Scan.cpp

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) Set LM9832 to idle (register 7 = 0x00) Read x lines of data Repeat y times until all of the calibration area has been scanned Generate average values for each pixel from x*y lines 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			

Function Performed	Function Explanation (What is done, why?)	Function Details	Related Software Modules
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 applica-tion, or file as selected. Scan of image data done at user set resolutions, color depth and window size/loca-tion. Scanning is complete when the requested # of lines have been transmitted back to the PC.		ScanSome() in Scan.cpp
Home Sensor			

Table 7: Scanner System Performance Limitations

Resolution	Line Size	PC I/O Speed	**Limit Reached** /Discussion	Potential Solutions
High	Full	Fast	Integration Time Limited by DRAM speed in LM9832.	LM9832 use Int Time Adj, Lamp Intensity Use Line Rate Mode to reduce integration time by factor of 3.
High	Full	Slow	Integration Time Limited by DRAM speed in LM9832. Data generation rate >> than PC I/O rate. Pausing required to meet constraints.	LM9832 use Int Time Adj, Lamp Intensity Use Line Rate Mode to reduce integration time by factor of 3. LM9830/32 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 LM9832.	LM9832 use Int Time Adj, Lamp Intensity
High	Partial	Fast	**Motor speed** limits scan speed Max 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 Int time OK, motor can't go fast enough.	Set integration time to maximum, increase VDPI 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. [Rev 2.5 + Add a separate counter for # of lines scanned in future Merlin]

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 8: System/Hardware Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please the LM9831 datasheet for more information, especially for Direct register parameters.)
Crystal Frequency	Indirect	08,
DRAM Size	Direct	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 good 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

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please the LM9831 datasheet for more information, especially for Direct register parameters.)
Sensor control signals active/disabled	Direct	0D
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/tim-ing	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 9: User Defined Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please the LM9831 datasheet 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	
Color or Grayscale	Direct	

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please the LM9831 datasheet for more information, especially for Direct register parameters.)
Bit depth (1,2,4,8,...) bits per pixel/color	Direct	
Tone mapping curve (Gamma)/thresholding	Direct	Gamma Table(s),
Preview or Full scan		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 10: Measured Parameters

Parameter Name	Direct or Indirect Affect on Register Settings	Registers Affected (Please the LM9831 datasheet for more information, especially for Direct register parameters.)
Host-Scanner datalink bandwidth		
Coarse calibration data		
Fine calibration data		
Sensor saturation point/maximum integration time		

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 remaining scanner settings. Scanner settings will include both LM9831 register settings, and software settings used in processing scan data. First we will consider all of the register settings. (Any registers omitted from the list are reserved.) Please refer to “Procedure for Adjusting Register Settings” on page 19 for more information on loading configuration registers.

5.4.1 Register 0 (Pixel Data)

The pixel (image) data is read from this register

5.4.2 Register 1 (Pixel Data Buffer status)

This register tells you how much image data is in the DRAM buffer.

5.4.3 Register 2 (Paper Sensor 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.

5.4.4 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.

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

Bit 5 is the read-only Powerdrop bit. It is set if the supply voltage drops below 3V. It is cleared by reading register 3. Note: this bit may be intermittent. It is currently recommended that this function not be used.

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 x 16 or a 1M x 16 DRAM is installed in a system. <insert instructions for DRAM test mode here>

5.4.5 Registers 4,5,6 (DataPort Address and Data)

These registers are used for selecting and accessing the gain, offset and gamma coefficient tables in RAM.

5.4.6 Register 7

This is the main “Control” register in the LM9832/LM9833. Please refer to “Procedure for Adjusting Register Settings” on page 19 for more information on adjusting Register 7 and loading configuration settings.

5.4.7 Register 8 (MCLK Divider)

This controls 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 as the USB interface can handle (typically 800kbytes/s).

DRAM speed limitations require that the following condition always be met:

MCLK_Divider * HDPI_Divider * Int_Time_Adjust >= 6.

5.4.8 Register 9 D2-D0 (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_Divider * HDPI_Divider * Int_Time_Adjust >= 6.

5.4.9 Register 9 D4-D3 (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 grayscale scans. Line art mode typically uses 1 bit per pixel (the threshold for line art mode can be set by the gamma table).

5.4.10 Register 9 D5 (DataMode)

This controls whether the pixel output data will be 8 bit (or less, determined by the Pixel Packing setting) or 14/16 bit. 14/16 bit mode is used for calibration. (The LM9832 provides 14 bit left justified data with the two LSBs equal to zero, while the LM9833 provides 16 bit data).

5.4.11 Register 9 D7-D6 (Chip Analog Bias Current - Power Reduction)

The analog bias current for the IC can be adjusted from 50% to 100% of normal. The recommended setting is 80% to provide the best combination of performance and power consumption. Lower settings will provide lower power consumption, but may reduce the ADC INL and DNL performance.

5.4.12 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. See the datasheet and “Sensor Timing Generation” on page 11 for more information.

5.4.13 Register 19 (Integration Time Adjust)

When this register is set to a non-zero value, the TR period alternates between 2 timebases to accommodate integration times shorter than those allowed by the 1MHz maximum data rate of the ADC. The short t_{INT} (t_{INTS}) is the same length as it was before the Int_Time_Adj, and the long t_{INT} (t_{INTL}) becomes equal to Int_Time_Adj * t_{INTS}. The longer integration time allows the DRAM to write pixel data to RAM.

DRAM speed limitations require that the following condition always be met:

MCLK_Divider * HDPI_Divider * Int_Time_Adjust >= 6.

The ITA function assumes that every second integration period is longer than the normal acquisition integration period. The charge accumulated during the longer period is clocked out of the CCD but the pixel values are not converted. This concept works well if the CCD used has a low “Image Lag” specification. This specification measures how much residual charge remains in the pixels after they have been clocked out the first time. If a CCD has a high Image Lag spec., then residual charge from the long integration time will interfere with the charge accumulated during the normal acquisition integration period. This can cause significant image degradation, especially if the long integration period is long enough to saturate the pixels.

5.4.14 Registers 1A and 1B (Stepper Phase Correction)

This register is used to control the synchronization between the Stepper Motor steps and the sensor integration period. For optimum image sharpness, the motor step should not occur during the image integration. This is especially important when scanning at the highest possible resolution settings. At lower resolutions, multiple steps are taken during each integration period, and an inherent image averaging occurs. Use of these registers is linked to proper selection of the Scanning Step Size and the Pixels per Line values. If the motor steps and Integration Periods are not of identical frequency and phase locked, the Stepper Phase Correction setting is irrelevant. If unsynchronized scanning is being done, this register should be set to 1. Please refer to the LM9832/LM9833 software package for additional details related to calculating the Scanning Step Size, Pixels Per Line, and Stepper Phase Correction values.

5.4.15 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 charge the DC blocking 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 allows more time for charging of the DC blocking capacitor.

5.4.16 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.

5.4.17 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. Set to the same value as Data Pixels Start in Registers 22 and 23.

5.4.18 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 corrupted image data.

The best image quality is achieved when the stepper motor steps are synchronized to the sensor integration period. Motor steps taking during the integration period will reduce the sharpness of the image. To provide optimal image quality, the Line End value is calculated to give a stepper motor rate that is exactly synchronized in frequency to the integration period. In addition, the Stepper Phase Correction value in Registers 1A and 1B should be set to position the motor step to coincide with the TR pulse when the image is not being integrated. When CIS sensors are used, the motor step can be synchronized with a period when the LED illumination is turned off.

Please refer to the LM9832/LM9833 software package for additional details related to calculating the Scanning Step Size, Pixels Per Line, and Stepper Phase Correction values.

5.4.19 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.

5.4.20 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.

(Data Pixels End - Data Pixels Start) should be an integer multiple of 2 x HDPI_Divider. This ensures that the pixel packing function will always pack the full number of pixel values for storage in DRAM and transmission to the host. Please refer to the LM9832/LM9833 software tools for more information.

5.4.21 Register 26 (Color Mode Settings)

This 8 bit register selects the proper mode of AFE operation for the type of sensor used, and the color mode desired by the user (color or monochrome). See the datasheet and “AFE and Sensor Control Signal Timing Modes” on page 14 for more information.

5.4.22 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. This can be useful to balance the different integrations to correspond to the differing sensitivities of the Red, Green and Blue pixel sensors. It is also useful when using transmissive rather than reflective media, such as slides or negatives. Since this type of media can have vastly different transmission efficiencies in the different wavelengths, extending the integration time for selective colours can help enhance the signal to noise ratio on the colors with less light transmission.

5.4.23 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 during subsequent operation.

5.4.24 Registers 2A through 37 (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).

5.4.25 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.

5.4.26 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. These registers should be loaded with 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 when the scanner is initially connected to the system. Please refer to “Scanning and Calibration” on page 25 for more information.

5.4.27 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 (or during) 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.

5.4.28 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.

5.4.29 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.

5.4.30 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.

5.4.31 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.

Bit 4 controls the state of the output pin drivers for the motor control outputs. Set to 0 by default to place the outputs in high impedance Tri-State mode. Ensure that pull-down or pull-up resistors are installed to force drive circuitry into a safe state when these outputs are in Tri-State.

Bit 5 controls the Line Skipping Phase for the n out of m function.

5.4.32 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:

stepsize = line_length * VDPI / (FSPI * 4), where:

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

line_length = 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 line_length.)

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

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:

pixel_frequency = 48MHz / (8 * C * MCLK_DIV), where:

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

8 = # MCLKs per pixel

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

MCLK_DIV = 1 + <reg 08>/2

pulses per second = steps per second = pixel_frequency/stepsize

So to calculate pulses per second, if a “pulse” = 1 fullstep, then:

pps = 48MHz / (8 * C * MCLK_DIV * 4 * stepsize)

If a “pulse” = 1 microstep, then

pps = 48MHz / (8 * C * MCLK_DIV * stepsize)

5.4.33 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 somewhat lower than the hardware maximum is used. Frequently this value will be entered in ‘.ini’ file settings to allow easy changes as software and hardware are developed independently.

5.4.34 XRegisters 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).

5.4.35 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).

5.4.36 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 LM9832/3 continues scanning until the current line is finished, then pauses. Register 4E must be set to allow this final line to finish scanning without 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 “LineDataSize (Bytes) Exact line data size for LM9832/3” on page 45 for details on calculation of LineDataSize.

5.4.37 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). This

value should be set so that the buffer is almost, but not quite empty when data starts being stored after the resume has been issued. This is dependent on several different parameters including; “Fullsteps to reverse when paused”, the average data transfer speed over USB, and the current acceleration profile settings.

5.4.38 Register 50 (Full Steps to Reverse When Paused)

The value in this register determines how many full steps (0 to 255) 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.

5.4.39 Register 51 D7-D3 (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. When accelerating or decelerating, 0, 1, 2, or 8 full steps can be taken at 0%, 25% and 50% speeds.

5.4.40 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.

5.4.41 Register 54 D2-D0 (Lines to Process After Pause/Lines to Discard After Resume)

The three LSBs 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.

5.4.42 Register 54 D7-D3 (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.

5.4.43 Register 55 D2-D0 (Kickstart Steps - Fullstepping Mode)

When fullstepping, bits 0 to 2 are used to set the number of full steps (0 to 7) where maximum current (0.465V/Rsense) is applied to 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.

5.4.44 Register 55 D7-D3 (Hold Current Timeout, DO NOT SET TO ZERO)

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 require-

ments 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. DO NOT SET TO ZERO.

5.4.45 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.

OSC = 48 MHz

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 typical initial setting for most small stepper motors and drive systems is n = 8.

5.4.46 Register 57 (Stepper Motor Minimum PWM Duty Cycle)

This register sets the minimum PWM duty cycle from 0% to 98% (0/64 to 63/64).

The LM9832/3 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.

5.4.47 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). PAPER SENSE 2 is also used in conjunction with the Step Counter function for programmed scanning and fast feed operations. See the LM9832/3 datasheets for additional information. The settings for this register will frequently be included in the '.ini' file for ease of adjustment during development.

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

These registers are used to configure the 6 MISC I/O pins on the LM9832/3. 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 LM9832/3 has been configured by software. These pins can be used to sense multiple input switches and pushbuttons, and to illuminate LEDs etc.

5.4.49 Registers 5C and 5D (ADC Output Code to Pixel Processing)

These registers are used during product or system testing to send a known fixed value to the HDPI section of pixel processing. Register 5E is used to enable or disable this mode.

5.4.50 Register 5E (ADC Test Mode - Set to 00h for normal operation)

This register is used to configure various test modes for device and system testing. Please refer to the product datasheet for additional information. The most useful feature to customers is the CDS Signal

output. By setting Bit 7 = 1, the CDS timing signal is output on the LampB pin. This allows the timing relationship between the sensor control signals, sensor output signal and LM9832/3 input sampling to be observed using an oscilloscope. The input sampling operation captures the Signal Reference on the rising edge of the CDS Signal and the Signal Value on the falling edge of the CDS Signal.

5.4.51 Register 69 (Version Number)

XXXXX011 = LM9831

XXXXX100 = LM9832/3

6. 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** (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 LM9832/3

$$\text{LineDataSize} = 2 \times \text{INT}\left(\frac{\text{INT}\left(\frac{\text{DataPixelsEnd} - \text{DataPixelsStart}}{\text{HDPI\_ADJ}}\right) \times \text{CM} \times \text{BitsPerPixel}}{16}\right)$$

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)

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

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

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

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

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

6.0.1 Symbol Definitions

StepSize - Pixels/Microstep. Number of pixels per microstep of stepper motor
LineLength - Pixels. Number of pixels between TR (CCD or CIS TRansfer) pulses
DataPixelsEnd - Pixels. Last valid data pixel in a line
DataPixelsStart - Pixels. First valid data pixel in a line
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)
t_{INT} - Integration time which is the time between two TR pulses
BytesPerSecond - Bytes/Second. Number of bytes of image data generated per second, includes effects of HDPI adjust and PP (pixel packing)
StepsPerSecond - Fullsteps/Second. Number of full steps per second or MicroStepsPerSecond/4
MicroStepsPerSecond - Microsteps/Second. Number of microsteps per second
FSPI - Full Steps/Inch, how many motor full steps are required to move one inch down the page
VDPI - Vertical Dot Per Inch, Dots/Inch or Pixels/Inch, describes vertical scanning resolution
HDPI_ADJ - Horizontal Dot Per Inch Adjustment unitless, used to reduce horizontal scanning resolution
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.
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.
CM - Color Mode, unitless. For 3 channel pixel rate color CM=3, for line rate color or gray scale scanning, CM=1
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

7. Code Fragments

The function ComputeDPD calculates the value for the DPD (Default Phase Difference) setting of registers 51, 52 and 53 and also returns the value of tr (number of pixels in one integration time).

```
//////////////////////////////////////////////////////////////////////////
//--------------------------------------------------------
// FUNCTION:ComputeDPD(int *regs,
//BOOL bWantCM /*=FALSE*/)

// DESCRIPTION:compute dpd reg 52:53

// INPUTS:regs- merlin registers
//bWantCm- TRUE if calculated tr contains cm component
//- FALSE if calculated tr does not include cm component

// RETURNS:tr- calculated tr
//--------------------------------------------------------
// bWantCM = TRUE by default, if bWantCM=0, then tr is returned with cm removed
int CHardware::ComputeDPD(int *regs,BOOL bWantCM /*=TRUE*/)
{
int linend;// line end value reg 20:21
int actpst;// active pixel start reg 1e:1f
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;// 3 if 3 channel line rate or 1 channel mode b
// 1 if pixel rate or 1 channel mode a
int afelat;// 4 if afeop=0, 7 otherwise
int b;// if ctmode=0, (ntr+1)*((2*gbnd)+dur+1), otherwise 1
int a;// if vps != 0 tcdp*(linend - actpst - afelat)/vps, otherwise 0
int tr;
//new following

//want tr/(4*stepsize) == integer if possible
//force TR to be multiple of 48 so aligned with
//standard 4*step sizes for /1,/2,/3,/4,/6,/12
int multiple = 48;
while (1)
{
// compute linend
int value = regs[0x21]; // line end lsb
```

```
linend = value;
value = regs[0x20]; // line end msb
linend += 256 * value;

// compute active pixel start
value = regs[0x1F]; // active pixel start lsb
actpst = value;
value = regs[0x1E]; // active pixel start msb
actpst += 256 * value;

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

tpsel = value & 3; // turbo/preview mode select

value = regs[0x0E]; // guardband/duration
gbnd = (value & 0xF0)/16;

dur = (value & 0xF);

value = regs[0x0D]; // number of tr pulses
ntr = value/128;

// 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;

value = regs[0x19]; // cis tr timing mode
int tradj;
tradj = value & 0x7F;
value = regs[0xb]/8;

ctmode = value & 3;

value = regs[0x51]; // quarter speed count
value /=4;
qtcnt = (value & 0xC)/4;
hfcnt = (value & 0x30)/16; // half speed count
if (LM9832() || LM9833())
{
if (qtcnt == 3) qtcnt = 8;
if (hfcnt == 3) hfcnt = 8;
}

value = regs[0x50]; // steps to reverse
strev = value & 0x3F;

value = regs[0x47]; // scan step size lsb
st = value;
value = regs[0x46]; // scan step size msb
st += 256 * (int) value;

//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

afelat = 7;
if (afeop == 0) afelat = 4;

if (tpsel == 0) tp = 1;
else
{
tp = tpspd +2;
if (tp==5) tp++;
}

b = 1;
if (ctmode == 0)
{
b = (ntr+1)*((2*gbnd)+dur+1);
b+= (1-ntr)*en_tradj;
}
if (ctmode == 2) b = 3;

a = linend;

tr = cm * (a + tp*(b + 3 - ntr));

{
//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%multiple)==0)
break;
linend++;
regs[0x20] = linend/256;
regs[0x21] = linend&0xff;
}

//dpd
if (tr == 0) dpd = 0;
else {
dpd = (((qtcnt*4) + (hfcnt*2) + strev)*4*st) % tr;
dpd = tr - dpd;
}

if (!bWantCM && cm != 0) tr /= cm; // remove cm component if requested

regs[0x53] = dpd & 0xFF;
regs[0x52] = (dpd>>8) & 0xFF;
regs[0x51] |= ((dpd>>16) & 3);

return tr;
}
```

8. USB Interface

This document describes the host's view of the USB LM9831, the logic between the LM9831 and the USB interface and any other points specific to the use of the USB interface in LM9831 not described in the Configurable USB Device Adapter documentation.

The device implements one vendor specific USB interface, which consists of the control endpoint plus 3 other endpoints.

The last section is addendum documenting the changes between the LM9831 and the LM9832 and LM9833.

8.1 Endpoints

The USB interface supports 4 endpoints.

8.1.1 Control Endpoint

Endpoint number is 0, maximum packet size is 8 bytes, single buffering is used.

8.1.1.1 Requests

The default control endpoint supports the required standard requests.

Two additional vendor specific interface requests are available to read and write registers. Each request reads or writes a series of registers, beginning at the first register specified.

The WRITE_CONTROL request is patterned after the proposed imaging class specific interface request of the same name. For this request, bmRequestType = 0x41, bRequest = 0x00, wValue = address of first register to be written (0x0000 to 0x00ff), wIndex = 0x0000, and wLength = number of registers to be written (0x0000 to 0x00c0), and data is the data to be written.

The READ_CONTROL request is patterned after the proposed imaging class specific interface request of the same name. For this request, bmRequestType = 0xc1, bRequest = 0x00, wValue = address of first register to be read (0x0000 to 0x00ff), wIndex = 0x0000, and wLength = number of registers to be read (0x0000 to 0x00c0), and data is the data read.

8.1.1.2 Descriptors

The device descriptor programmed into the internal ROM specifies the following:

bLength = 0x12
bDescriptorType = 0x01 = DEVICE
bcdUSB[7:0] = 0x00 = 1.00
bcdUSB[15:8] = 0x01 = 1.00
bDeviceClass = 0x00 = Independent class interfaces
bDeviceSubClass = 0x00
bDeviceProtocol = 0x00
bMaxPacketSize0 = 0x08
idVendor[7:0] = 0x00 = National Semiconductor
idVendor[15:8] = 0x04 = National Semiconductor
idProduct[7:0] = 0x00
idProduct[15:8] = 0x10
bcdDevice[7:0] = 0x00 = 1.00
bcdDevice[15:8] = 0x01 = 1.00
iManufacturer = 0x01
iProduct = 0x02
iSerialNumber = 0x00 = not specified
bNumConfigurations = 0x01

The configuration descriptor programmed into the internal ROM specifies the following:
bLength = 0x09
bDescriptorType = 0x02 = CONFIGURATION
wTotalLength[7:0] = 0x27
wTotalLength[15:8] = 0x00
bNumInterfaces = 0x01
bConfigurationValue = 0x01
iConfiguration = 0x00 = not specified
bmAttributes = 0xa0 = {BusPowered,RemoteWakeup} if self_powered pin is low
= 0x60 = {SelfPowered,RemoteWakeup} if self_powered pin is high
MaxPower = 0xfa = 500ma if self_powered pin is low
= 0x01 = 2ma if self_powered pin is high
The interface descriptor programmed into the internal ROM specifies the following:
bLength = 0x09
bDescriptorType = 0x04 = INTERFACE
bInterfaceNumber = 0x00
bAlternateSetting = 0x00
bNumEndpoints = 0x03
bInterfaceClass = 0xff = Vendor
bInterfaceSubClass = 0x00
bInterfaceProtocol = 0xff = Vendor
iInterface = 0x00 = not specified
The endpoint 1 descriptor programmed into the internal ROM specifies the following:
bLength = 0x07
bDescriptorType = 0x05 = ENDPOINT
bEndpointAddress = 0x81 = {IN,1}
bmAttributes = 0x03 = Interrupt
wMaxPacketSize[7:0] = 0x01
wMaxPacketSize[15:8] = 0x00
bInterval = 0x10 = 16ms
The endpoint 2 descriptor programmed into the internal ROM specifies the following:
bLength = 0x07
bDescriptorType = 0x05 = ENDPOINT
bEndpointAddress = 0x82 = {IN,2}
bmAttributes = 0x02 = Bulk TBD: is this bit encoding right?
wMaxPacketSize[7:0] = 0x40
wMaxPacketSize[15:8] = 0x00
bInterval = 0x00
The endpoint 3 descriptor programmed into the internal ROM specifies the following:
bLength = 0x07
bDescriptorType = 0x05 = ENDPOINT
bEndpointAddress = 0x03 = {OUT,3}
bmAttributes = 0x02 = Bulk TBD: is this bit encoding right?
wMaxPacketSize[7:0] = 0x40
wMaxPacketSize[15:8] = 0x00
bInterval = 0x00
String descriptor 0 programmed into the internal ROM specifies the language IDs supported:
bLength = 0x04
bDescriptorType = 0x03 = STRING
LangID[7:0] = 0x09 = Primary Language = English
LangID[15:8] = 0x04 = Sub Language = US
String descriptor 1 programmed into the internal ROM specifies the manufacturer (in UNICODE):
bLength = 0x2e

bDescriptorType = 0x03 = STRING
string = “National Semiconductor”
String descriptor 2 programmed into the internal ROM specifies the product (in UNICODE):
bLength = 0x1e
bDescriptorType = 0x03 = STRING
string = “LM9832 Scanner”

8.1.2 Interrupt Endpoint

Endpoint number is 1, maximum packet size is 1 byte, buffer size is 8 bytes, single buffering is used.

Single data byte packets are returned on this endpoint. When set, each bit of the data byte indicates that the corresponding bit of the miscellaneous I/O status register (0x02) has changed since the register was last read. Such packets are only returned when the status bits change. When the host receives such a packet, it should then read the status register to determine the current values.

8.1.3 Bulk In Endpoint

Endpoint number is 2, maximum packet size is 64 bytes, double buffering is used, the buffer is shared with the bulk out endpoint.

Registers may be read via the bulk in endpoint by sending a sequence of command bytes on the bulk out endpoint, then reading the data bytes on the bulk in endpoint. There are 4 command bytes, which, in order, indicate the mode of the transfer, the starting address, and the number of bytes to be read (most significant byte, followed by least significant byte). The mode byte is bit mapped: bit 0 is set to indicate a register read operation; bit 1, if set, indicates that each byte will be read from the register at the next higher address that the previous byte (incrementing address mode), if clear, indicates that each byte will be read from the same register; all other bits are cleared.

Because register reads via this endpoint share the bulk out endpoint with register writes, such a write must complete before such a read is started, or vice versa. Since reads and writes via the control endpoint use different resources, reads or writes on the control endpoint may be performed simultaneously with reads or writes on the bulk endpoints.

8.1.4 Bulk Out Endpoint

Endpoint number is 3, maximum packet size is 64 bytes, double buffering is used, the buffer is shared with the bulk in endpoint.

Registers may be written via the bulk out endpoint by sending a sequence of command bytes, followed by the data bytes. There are 4 command bytes, which, in order, indicate the mode of the transfer, the starting address, and the number of bytes to be written (most significant byte, followed by least significant byte). The mode byte is bit mapped: bit 0 is cleared to indicate a register write; bit 1, if set, indicates that each byte will be written to the register at the next higher address that the previous byte (incrementing address mode), if clear, indicates that each byte will be written to the same register; all other bits are cleared.

Because register writes via this endpoint share the endpoint with register writes, such a read must complete before such a write is started, or vice versa. Since reads and writes via the control endpoint use different resources, reads or writes on the control endpoint may be performed simultaneously with reads or writes on the bulk endpoints.

8.2 Registers

8.2.1 Address space

The address space available for register access is 0x00 to 0xbf. 0xc0 to 0xff is not available for registers because the data transfer interface uses this range to address endpoint pipes.

8.2.2 Operational registers

LM9831's existing registers are mapped directly into the 0x00 to 0x7f address range.

Non-blocking flow control is implemented on accesses to register 0x00 (Pixel Data). If a read cannot be performed on this register because data is not available in the buffer, a retry status will be returned to the USB interface, thus enabling other endpoints to read/write other registers.

Accesses to all other registers use blocking flow control. An acknowledgement is not returned to the USB interface until the data has been accepted or provided by LM9831.

8.2.3 Diagnostic registers

The following registers are available for diagnostic and production test purposes.

0xb6, frame0_diag, r/w, resets to 0x00, controls short frame test mode used for speeding up testing of suspend and resume timing, 0x00 enables normal frame timer length of 36015 cycles of the 12MHz clock, 0x01 enables short frame timer length of 2223 cycles of the 12MHz clock.

0xb7, mac0_diag, r/o, provides visibility to various mac states

0xb8, mac1_diag, r/o, provides visibility to various mac states

0xb9, mac2_diag, r/o, provides visibility to various mac states

0xba, mac3_diag, r/o, provides visibility to various mac states

0xbb, phy0_diag, r/o, provides visibility to various phy states

0xbc, xcvr0_diag, r/o, receiver test register, bit 0 samples the state of the differential receiver, bit 1 samples the state of the D- single ended receiver, bit 2 samples the state of the D+ single ended receiver. When the self_powered strap input is high, these bits reflect the current state of the receivers. When the self powered strap input is low, these bits are latched. To test the receivers, with the self_powered strap input high, drive the D+ and D- inputs with the desired levels; take the self_powered pin low to latch the sample; do a USB transfer to read this register. There are two sets of receivers, differential receivers which are active in the operational states, and CMOS receivers which are active in suspend state. Be careful not to allow the USB interface to enter the suspend state (bus in J state for 3ms or longer) when testing the differential receivers. Note that the clock need not be running to sample the receiver states.

0xbd, xcvr1_diag, r/w, resets to 0x00, transmitter D+ test sequence, see description below

0xbe, xcvr2_diag, r/w, resets to 0x00, transmitter D- test sequence, see description below

0xbf, xcvr3_diag, r/w, resets to 0x00, transmitter enable test sequence, when any bit of xcvr3_diag is set, and the self_powered strap input is high, the transmitter test mode is entered. In this mode, an 8 bit pattern is sent by the D+ and D- transmitters. For the first bit, xcvr1_diag[0] sets the D+ logic level, xcvr2_diag[0] sets the D- logic level, and xcvr3_diag[0], if set, enables both transmitters. For the second through eighth bits of the test sequence, bits 1 to 7 of these registers are likewise used. The sequence will repeat as long as self_powered is high.

8.3 Power management

The USB bus state and device state control the bus power consumed by the device. Both states are available to the LM9831 logic. If the USB is suspended, the device will draw no more than 500uA from the USB (the pull-up resistor consumes 200uA of this, nominally). If the USB is not suspended, but the device is not configured, the device will draw no more than 100mA from the USB. If the USB is not suspended and is configured, then the device may draw up to 500mA from the USB.

The miscellaneous I/O status register (0x02) forms the basis for initiating remote resume requests. If any bit of this register has changed since the last time the register was read, and the USB is suspended, a remote resume will be done. This logic is asynchronous to permit a remote resume to be initiated while the clock is stopped. Some of the logic may be shared with the interrupt endpoint logic. If the device remote wake-up enable feature is not enabled, the USB interface will not propagate the resume request onto the USB.

8.4 LM9832 Addendum

8.4.1 USB Interface ROM Changes from LM9831 to LM9832

There are two sets of changes. The first set adds support for vendor defined requests with the “device” as the recipient. The first ROM was coded to support vendor defined requests with only the “interface” as the recipient. Since then, Microsoft has released its Still Image interface (STI) for device drivers, which supports vendor specific requests with the “device” as the recipient, but not with the “interface” as the recipient. Furthermore, the STI does not even support the standard USB requests that enable and disable the device's remote wake-up feature. So, in order to support the READ_CONTROL, WRITE_CONTROL, SET_FEATURE(DEVICE_REMOTE_WAKEUP), and SET_FEATURE(DEVICE_REMOTE_WAKEUP) requests, vendor specific “device” versions of these requests were added to the ROM.

Incoming requests are interpreted by a progressive decision tree coded into the ROM. As each byte is read from the receive buffer, it is tested against several values. When a value matches, program execution branches to the code which further services that byte value. The new requests are supported by adding tests for the new byte values and adding corresponding code. Because only the first 2 bytes differ between the old and new READ_CONTROL and WRITE_CONTROL requests, this code merges back together, and the same code services byte 3 and onward of both the old and new requests.

The second set of changes consists of updates to the descriptors to indicate the new product, new revision of the USB specification, and use of vendor defined requests with the “device” as the recipient.

These changes were first coded into an external EEPROM and tested with several PC systems. The changes were then made to the uprog_sti.u file and compiled with the uasm script to produce a file for Verilog simulation and another file for ROM generation.

8.4.2 New Vendor-Specific Device Requests

bmReqTyp	bRequest	wValue	wIndex	wLength	Data	Description
40	04	00xx	0001	0001	yy	CLEAR_FEATURE(DEVICE_REMOTE_WAKEUP)
40	0C	00xx	0001	0001	yy	CLEAR_FEATURE(DEVICE_REMOTE_WAKEUP)
40	04	00xx	0003	0001	yy	SET_FEATURE(DEVICE_REMOTE_WAKEUP)

```
40      0C      00xx  0003  0001   yy  SET_FEATURE(DEVICE_REMOTE_WAKEUP)
40      04      offset  0000  length  data WRITE_CONTROL(register_offset,data)
40      0C      offset  0000  length  data WRITE_CONTROL(register_offset,data)
C0      04      offset  0000  length  data READ_CONTROL(register_offset)
C0      0C      offset  0000  length  data READ_CONTROL(register_offset)
```

xx = ignored (recommend setting to 01 to indicate DEVICE_REMOTE_WAKEUP feature selector)

yy = ignored, 1 byte

8.4.3 Changed Device Descriptor

```
bcdUSB:          from: 0x0100 = "1.0" to: 0x0101 = "1.1"
bDeviceClass:    from: 0x00   = Independent Class Interfaces to: 0xff   = Vendor Specific
bDeviceProtocol: from: 0x00 to: 0xff   = Vendor Specific
idProduct:       from: 0x1000 to: 0x1001
```

8.4.4 Changed Product String Descriptor

```
from: "Merlin Scanner" to: "LM9832 42 Bit Scanner"
```

8.4.5 Additional Notes

With USB 1.1, bit 7 of the bmAttributes byte of the Configuration Descriptor is now a reserved bit, and should always be set to 1. However, this change did not make it into this ROM, which uses the USB 1.0 definition of this bit, which indicates Bus Powered capability. Future revisions of the ROM should implement this change. Specifically, this involves making the following change in uprog.u:

```
from:    GetDescConfig1:  xmiti(0x60);          // bmAttributes = {SelfPowered,RemoteWakeup}
to:      GetDescConfig1:  xmiti(0xe0);          // bmAttributes = {SelfPowered,RemoteWakeup}
```

9. Support Tools

9.1 Software

There are two different types of software tools available.

“9832test.exe” is a test bench and evaluation program. It is useful for initial evaluation of the LM9832/LM9833 with the specific scanner hardware applicable in your system.

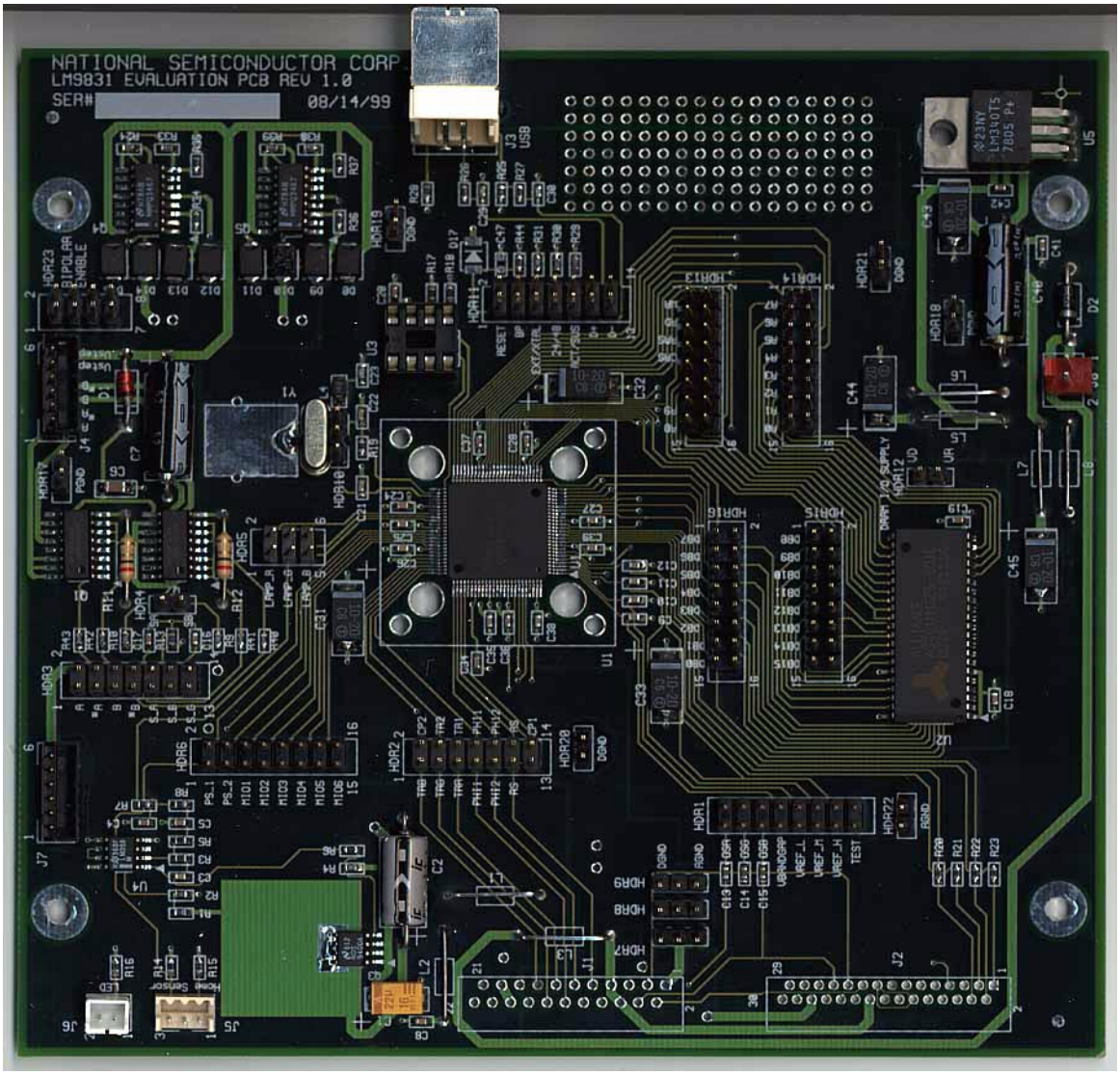
“LM983X.exe” is a TWAIN compliant software development package. It provides a starting point for development of final customer scanning software.

Full source code is available for both programs, however we highly recommend the LM983X package for ‘understanding’ the software requirements of the product.

9.2 Hardware

Evaluation boards are available for the LM9832 and LM9833 ICs. These boards are intended for evaluation of sensors and other customer hardware before the first circuit board is designed and fabricated. The only additional components required are the customer hardware including stepper motor, sensor, optics, etc.

Figure 18: LM9832/3 Evaluation Board.



A second alternative is to purchase a retail scanner which uses the LM9832 or LM9833. This product can be used to evaluate the performance of the LM9832 or LM9833 product and provide a tool for early software development. Contact your National Semiconductor support personnel for additional information regarding commercially available LM9832 or LM9833 based scanners.