820 Kifer Road
Sunnyvale, CA 94086
Main (408) 240-4700
Fax (408) 456-0708
www.siliconlight.com

“High-Speed Buddy” GLV Module Operating Manual

MNL-004827-01

Revision B

June 16, 2016

Document Status: Released

WARNING!

DAMAGE TO THE DEVICE MAY OCCUR IF MAXIMUM VOLTAGES ARE EXCEEDED!

HIGHER LASER POWER LEVELS RESULT IN LOWER VOLTAGE NEEDED FOR BOTH OPERATING ROLL-OVER AND MAXIMUM PERMISSABLE VOLTAGES.

FOR PIXEL VOLTAGE GUIDELINES AND RECOMMENDED SETUP PROCEDURE

SEE SECTIONS:

Section 3.1.7 Voltages that Drive the Pixel Ribbons,
Section 3.2.3.6 D/A Conversion Protocol &
Section 3.3.1 Power Up and Initialization Sequence

WARNING!

Table of Contents

Purpose: This operating manual presents basic information about the design and operation of the “High-Speed Buddy” Grating Light Valve (GLV) Module. This manual describes the operation of High-Speed Buddy Module and specifically describes the Grating Light Valve, GLV optics, physical description, electrical interface and data interface. The information is presented so that a qualified engineer or technician can gain a detailed understanding of the module design, functionality and system operation.

1 Grating Light Valve

1.1 Pixel Structure

The Grating Light Valve (GLV™) is a high-performance spatial light modulator composed of thousands of free-standing silicon-nitride micro-ribbons anchored on the surface of a silicon chip. By electronically controlling the deflection of the ribbons, the GLV functions as a programmable diffraction grating, enabling attenuation, modulation and switching of laser light with unparalleled resolution, speed, precision and reliability.

The GLV is a class of Micro-Electro-Mechanical Systems (MEMS) and is built from amorphous, stoichiometric silicon nitride (Si3N4) – a robust high-temperature ceramic well-known for its hardness, durability and chemical stability. Individual ribbons are between 2-4 um wide and 100-300um long. The ribbons are coated with a thin aluminum layer which serves both as a reflector and electrode. The ribbons are suspended over a lower “common” electrode. When a controlled voltage from a CMOS channel driver is delivered to the ribbon, it deflects toward the substrate in response to the electrostatic force. After removing the voltage, the ribbon returns to its un-deflected state by nature of the high intrinsic tensile stress of the silicon nitride.

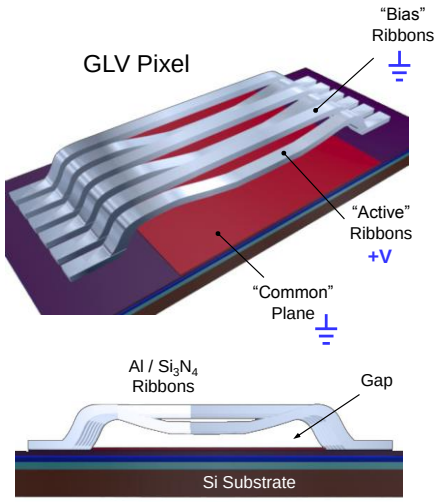

Figure 1-1 GLV Pixel structure

1.2 Operating Principle

Alternating GLV ribbons are arranged into two groups: moving “active” ribbons which are driven by an electronic driver channel and static “bias” ribbons which are typically grounded along with the common electrode. The active and bias ribbons are identical in every aspect except how they are driven. With no voltage applied to the active ribbons, they are co-planar with the bias ribbons. In this state, incident light onto the GLV is specularly reflected like a mirror. When a voltage is applied to the active ribbons, they deflect relative to the bias ribbons, establishing a square-well diffraction grating. In this state, incident light is diffracted into fixed diffraction angles. The amount of light reflected vs. diffracted can be continuously varied by controlling the voltage on the active ribbons (i.e. analog gray scale). The specular beam is fully extinguished when the active ribbons are displaced by a distance of ¼ of the wavelength relative to the bias ribbons.

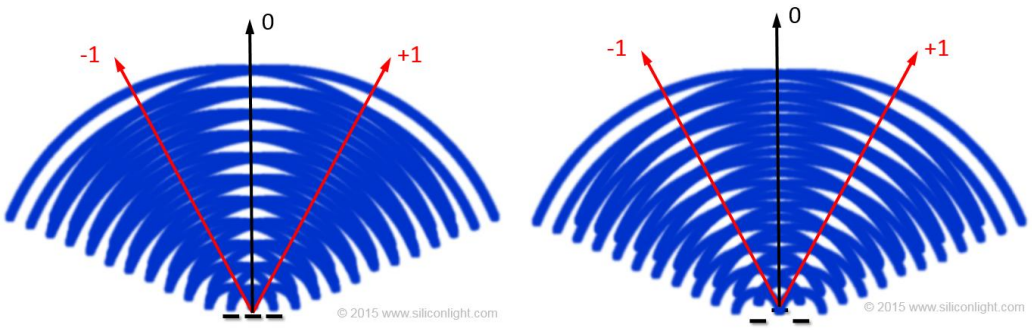

Figure 1-2 The GLV in (a) reflective and (b) diffractive states

1.3 Optical Response

The GLV has a continuous intensity-vs-voltage or “IV” response, as shown in Figure 1-3. 0th- and 1st-order IV curves are compliments of each other. The 0th-order response carries the highest net optical efficiency (typically 70-80%) and is favored in applications requiring high optical throughput. The GLV 1st-order response yields the highest contrast and contrast ratios in the millions have been achieved in this mode. The plot at right shows the dynamic (pulse) response of the GLV. The low mass, high tension and short stroke of the GLV ribbons allow it to switch at extremely high speeds. Typical transitions times are <300ns, enabling MHz modulation rates.

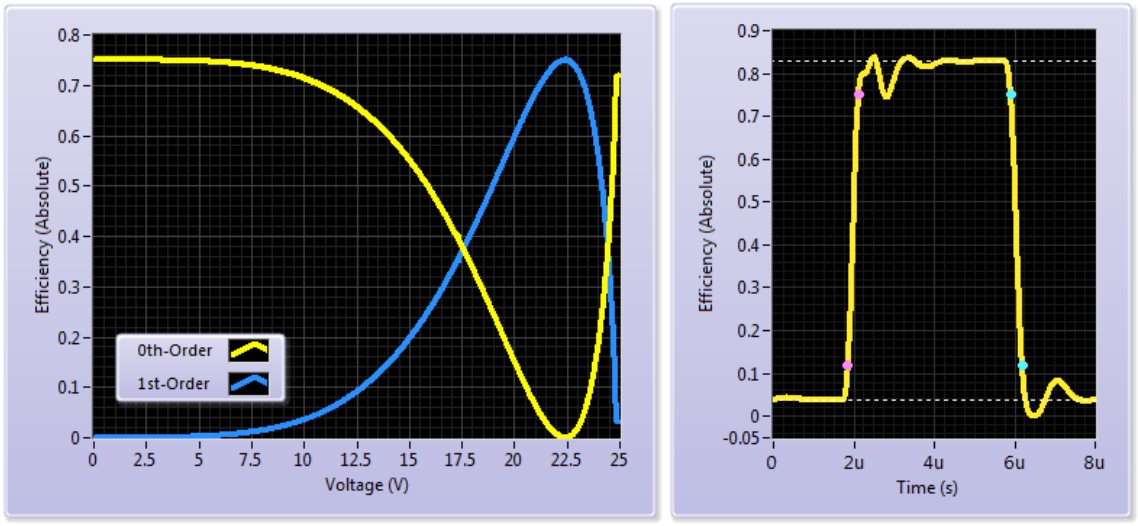

Figure 1-3 (left) GLV intensity-vs-voltage curves (0th and 1st order) and (right) GLV pulse

1.4 Pixel Array

In the High-Speed Buddy, the GLV is a one-dimensional array of 1088 pixels. Each pixel is composed of three active ribbons and three bias ribbons. Each ribbon is 3.75um and separated by 0.5um gaps. The pitch of the grating is determined by one active ribbon and one bias ribbon (including gaps). This dimension is 8.5um. Each pixel comprises three active-bias pairs for total pixel pitch of 25.5um along the GLV array axis. All three active ribbons are driven from a common pixel driver channel. The length of the ribbon is defined by the distance between post anchors and is 220um. The pixels at each end of the array (i.e. Pixel 1 and Pixel 1088) are special “dummy” pixels in which a larger number of active ribbons (36) are ganged together. These pixels are typically used to darken (i.e. not expose) regions near the ends of the array. The dummy pixels cannot be operated at full speed because the increased capacitance loads the driver.

1.5 GLV Optics

The GLV is a diffractive device and is used in conjunction with monochromatic light sources. Because of the GLV’s linear aspect ratio, circular laser illumination is first transformed into a line. Figure 1-4 shows a typical GLV illumination and imaging configuration. In this figure, the optical path has been unfolded around the GLV. Typically, the incident and reflected light paths are separated by imposing a small incidence angle to the illumination onto the GLV (typically 5 to 20° from surface normal). Alternatively, a polarization rotator and beam splitting optics can be used for normal incidence geometries.

Line illumination for the GLV can be generated from laser sources by a variety of methods including a Powell lens, “fly-eye” lens array, or simply by a cylindrical lens. In the latter case, the GLV itself can be used to attenuate or “flatten” the Gaussian light distribution along the GLV array axis. For illustration purposes, a Powell lens is shown in Figure 1-4. The circular beam from the laser is dispersed uniformly in angle in one axis. A “slow-axis” collimator is then used to collimate the rays into a “top-hat” profile of uniform intensity along the GLV array axis. Next, an orthogonal “fast-axis” cylindrical lens is used to focus the beam onto the GLV. Typical numerical apertures for the fast axis focus are between 0.01 and 0.1. The full width of the line illumination should be less than one third of the ribbon length (i.e. 50-75um) for best contrast. As explained earlier, the imaging system of the GLV comprises a conjugate lens pair with an aperture in the Fourier plane between the lenses, used to select the imaging order. Image magnification is determined by the ratio of the focal lengths of the lens pair (i.e., M = f2/f1).

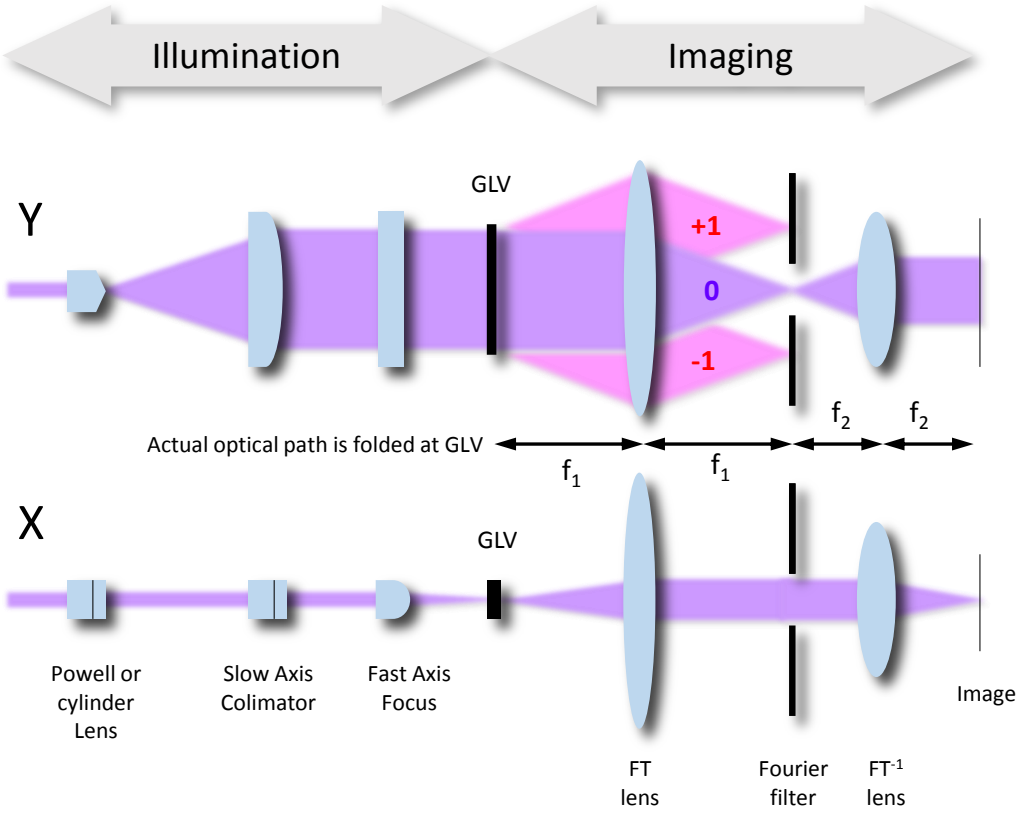

Figure 1-4 Top and side views of the GLV illumination and imaging optics

2 Physical Description

The “High-Speed Buddy” Module comes with a Cricket Interface Board and a custom 150mm flex cable to connect the two boards. This configuration is collectively known as the “High-Speed Buddy Set” and is illustrated in Figure 2-1. Options for the module set include a 300 mm version of the flex cable and a heat sink for the Cricket Interface Board.

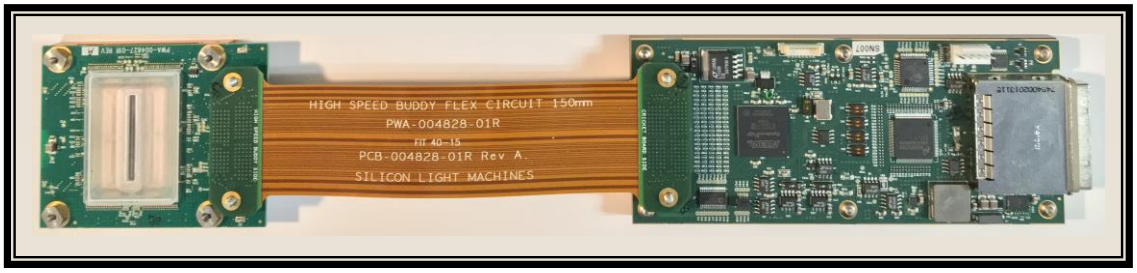

Figure 2-1 High-Speed Buddy Set
(High-Speed Buddy Module, Cricket Interface Board & Flex Cable)

Figure 2-2 shows a close up picture of the High-Speed Buddy Module, where the GLV in the middle surrounded by 4 SLM212 driver chips and enclosed with a slotted lid. A heat sink is mounted on the bottom. Four mounting fasteners are provided for mounting from the bottom.

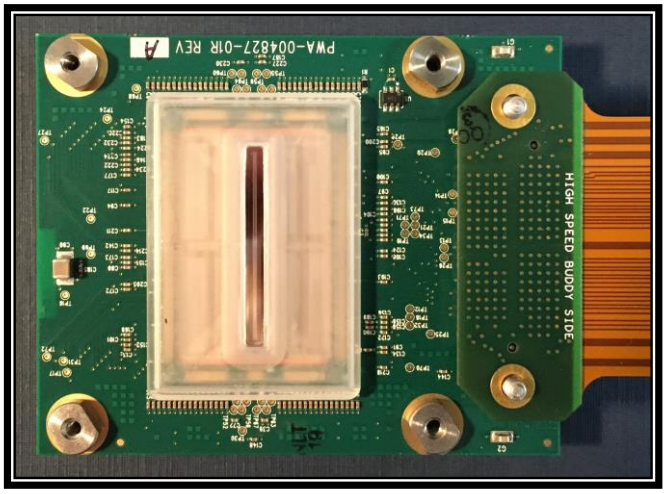

Figure 2-2 High-Speed Buddy Module

2.1 High-Speed Buddy Module – Mechanical Drawings

The High-Speed Buddy Module is a printed circuit board that contains the GLV Device, SLM212 drivers, input connector, and copper heat sink. Figure 2-3, Figure 2-4 and Figure 2-5 illustrate top, side and bottom views of the High-Speed Buddy Module. Four 8-32 threaded holes are provided on the bottom of the heat sink for mounting the module to the user's equipment. Optionally, two other sets of 4 mounting holes (4-40 threaded) are available. One 4-40 set allows mounting from the top and the second 4-40 set allows mounting from the bottom. Contact SLM for information on the locations of the optional 4-40 mounting holes.

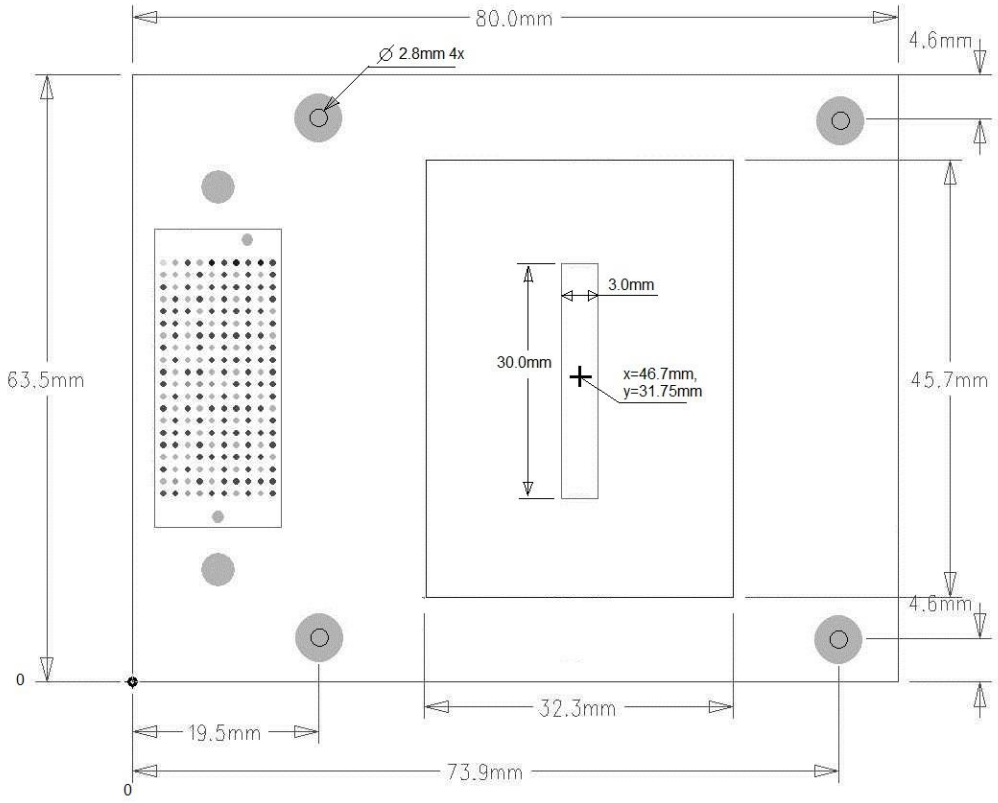

Figure 2-3 Top View – High-Speed Buddy Module

(All Dimensions are in mm with tolerances of ± 0.25 mm)

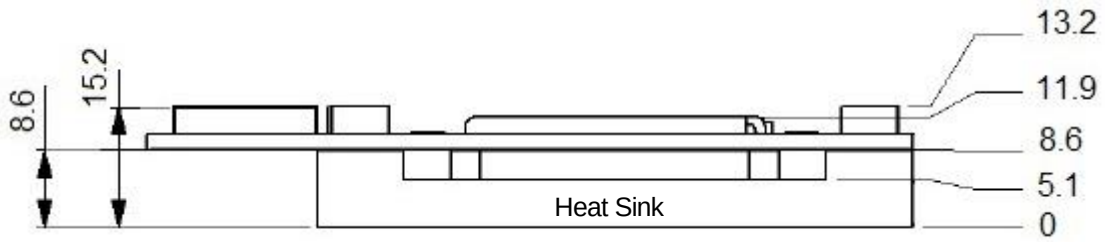

Figure 2-4 Side View – High-Speed Buddy Module

(All Dimensions are in mm with tolerances of ± 0.25 mm)

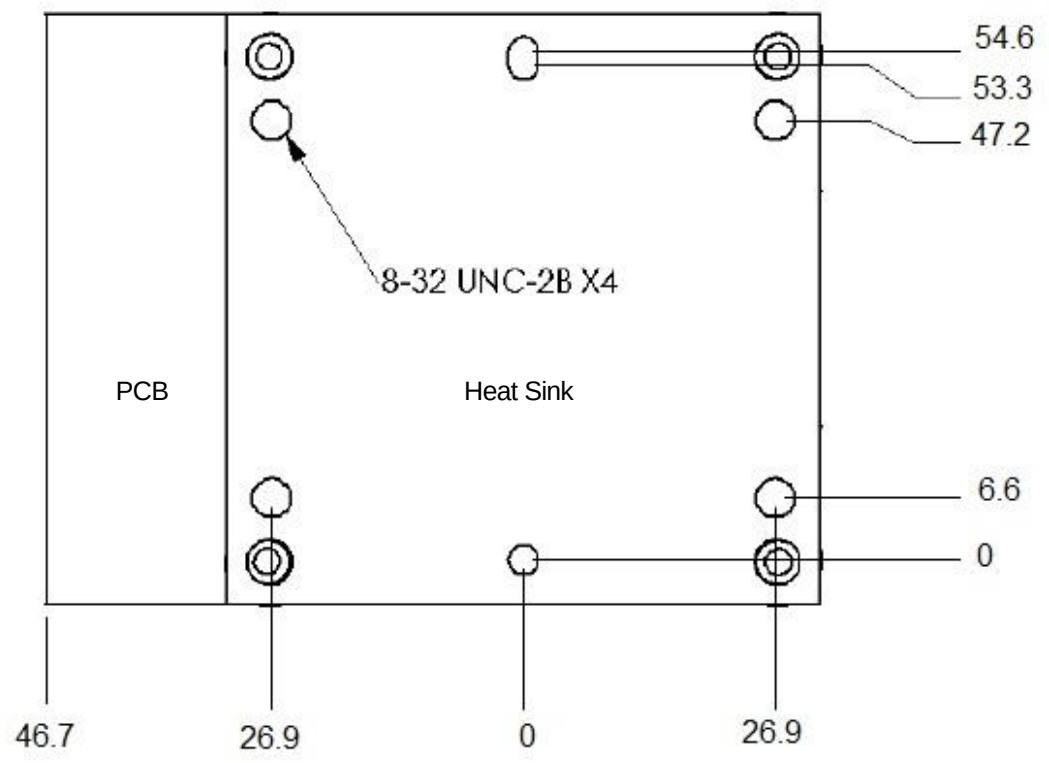

Figure 2-5 Bottom View– High-Speed Buddy Module

(All Dimensions are in mm with tolerances of ± 0.25 mm)

2.2 Cricket Interface Board – Mechanical Drawings

The Cricket Interface Board is a printed circuit board that contains the support electronics for the GLV and drivers, including a deserializer, FPGA, PSoC, power supply components and other miscellaneous components. Figure 2-6 and Figure 2-7 illustrate top and side views of the Cricket Interface Board. The side view shows the PCB stacked on top of the optional heat frame. Six M3 hole locations (3.45mm diameter) can be used to attach the bottom of the Cricket Board to the top of the optional heat frame or directly to the user equipment.

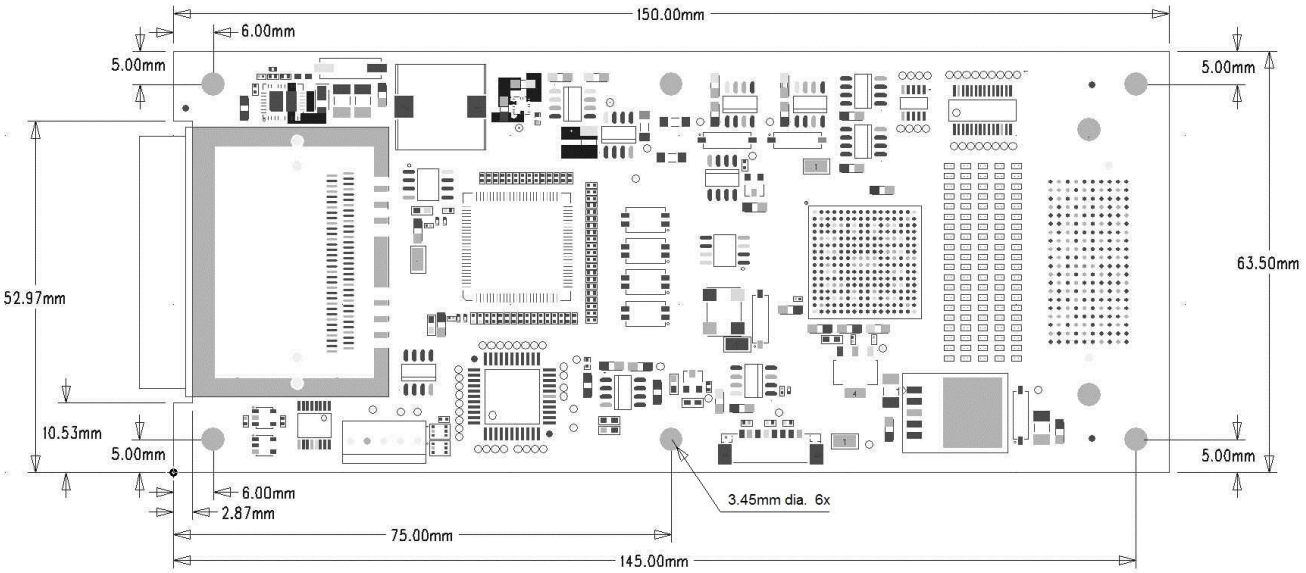

Figure 2-6 Top View– Cricket Interface Board

(All Dimensions are in mm with tolerances of ± 0.25 mm)

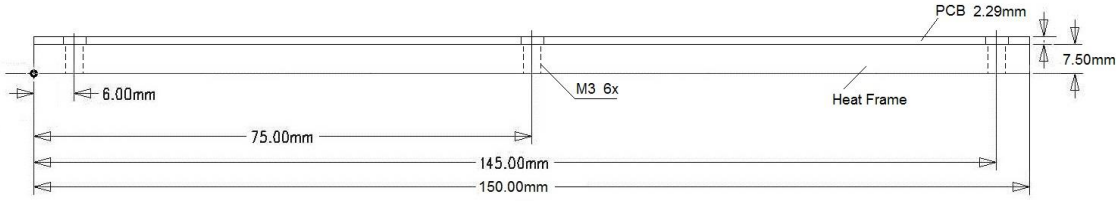

Figure 2-7 Side View– Cricket Interface Board

(All Dimensions are in mm with tolerances of ± 0.25 mm)

2.3 Cricket Heat Frame – Mechanical Drawings

The optional heat frame for the Cricket can be used to mount the Cricket Board to the user equipment (e.g. chiller plate, etc.). Figure 2-8 and Figure 2-9 illustrate the top view and side view of the Cricket heat frame. Six M3 hole locations are used to attach the bottom of the Cricket Board to the top of the heat frame. Eleven M4 tapped hole locations can be used to attach the bottom of the heat frame to the user equipment. The locations of all mounting holes are shows in the figures below.

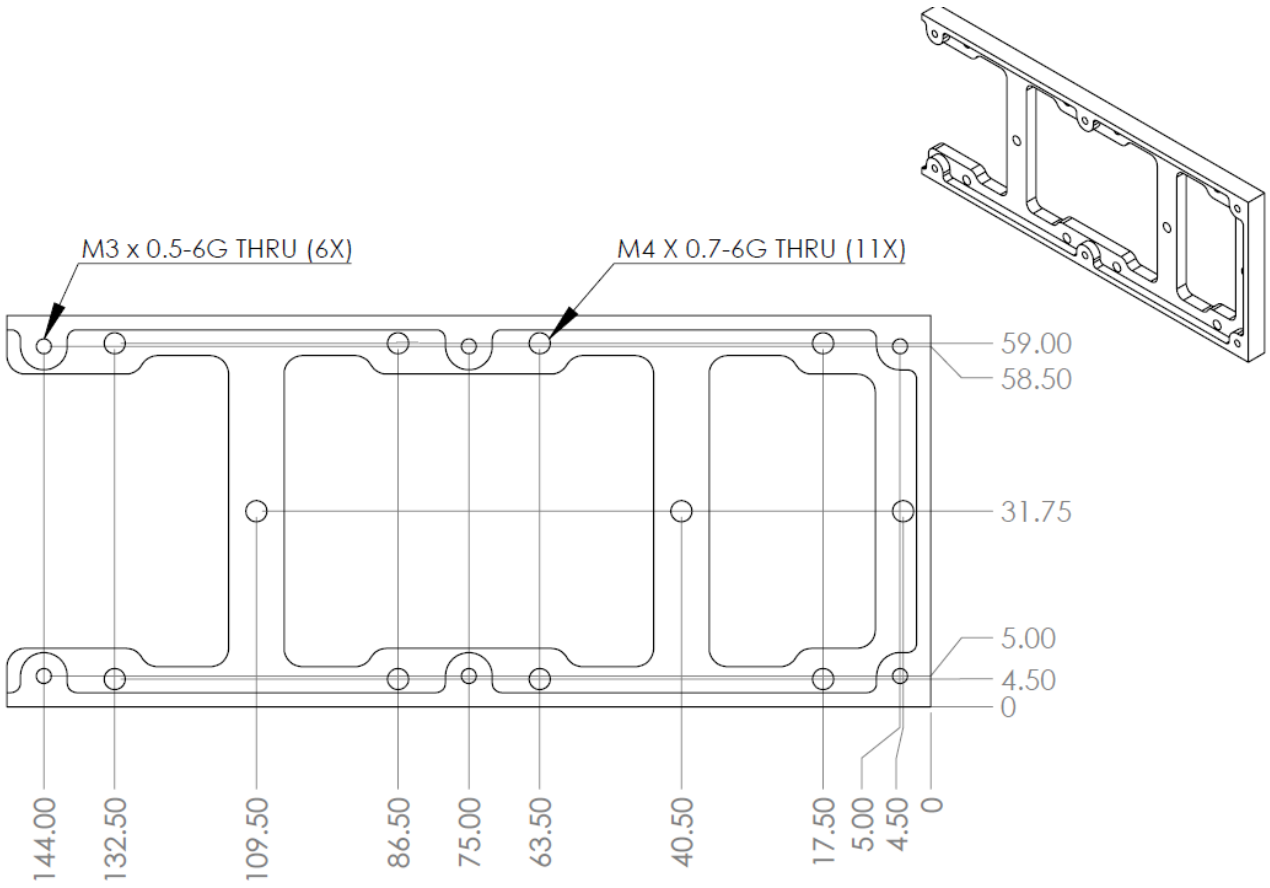

Figure 2-8 Top View - Cricket Heat Frame

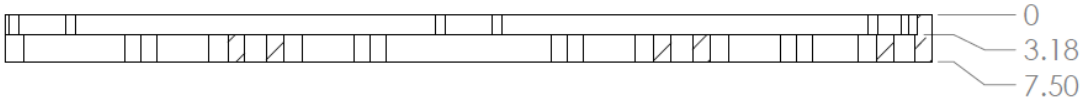

Figure 2-9 Side View – Cricket Heat Frame

3 Electrical Description

The high-voltages required to drive the GLV are supplied by a custom SLM212 Driver chip. Each SLM212 Driver chip has four 272 high-voltage Digital-to-Analog (DAC) channels. When the module is used within an appropriate optical system, each pixel can have 1024 different levels of intensity. This can be done with very high optical efficiency and excellent contrast.

The support electronics are implemented on a separate interface board. The interface board includes: LVDS deserializer, FPGA, Programmable System on a Chip (PSoC), health monitoring and power supply components. The LVDS deserializer provides an 8-channel high-speed data downlink to the module. The FPGA provides a data bridge from the LVDS deserializer to the 4 SLM212 driver chips. A programmable system on a chip (PSoC) device contains a processor that is controlled by the user with an I^2C bus. The PSoC provides for power sequencing, initialization and health monitoring. Figure 3-1 shows a block diagram for the High-Speed Buddy Module.

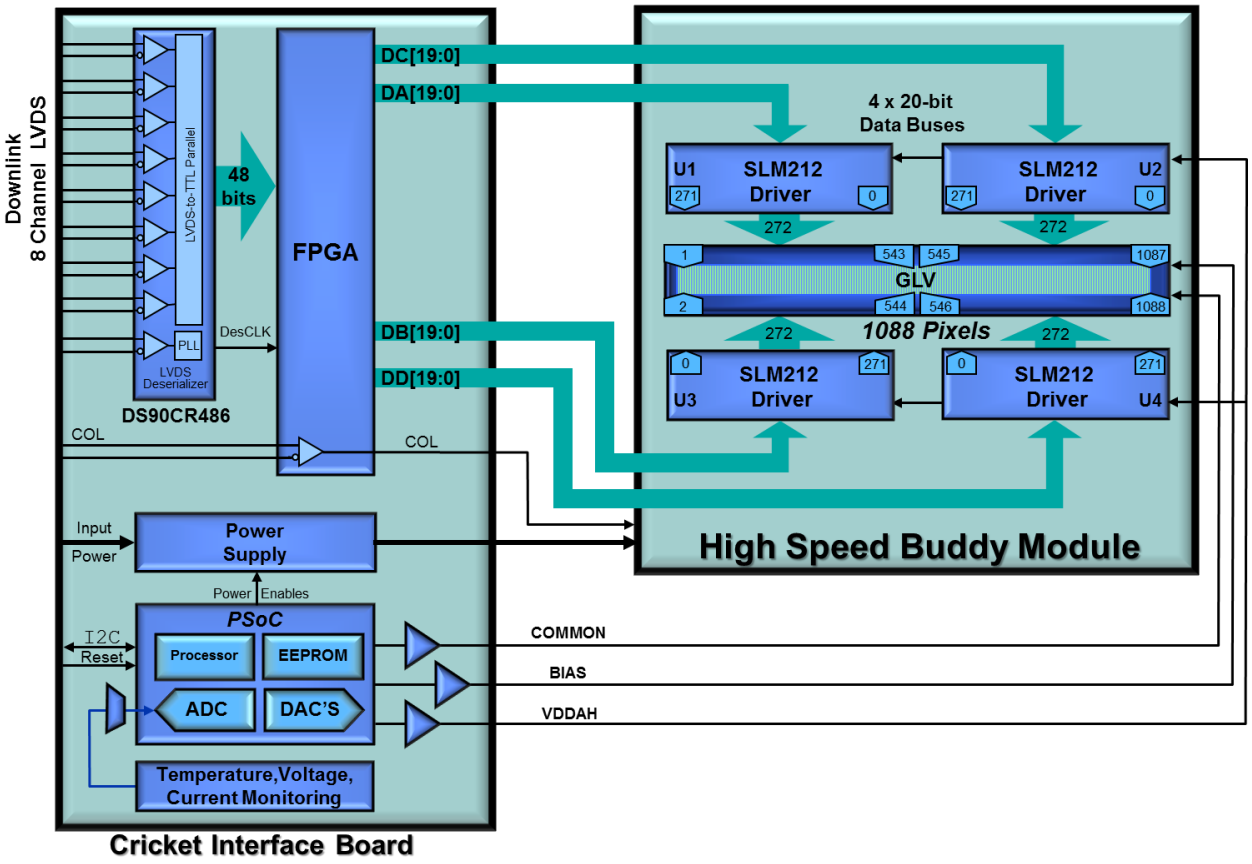

Figure 3-1 Module Block Diagram

Figure 3-2 shows the High-Speed Buddy Module connected to a typical control system. In the figure, the control board is the Cosmo Test Board which is part of the High-Speed Buddy evaluation kit. The Cosmo Board (or a customer controller board) provides imaging data, I^2C control signals, and a column strobe signals to the High-Speed Buddy Module over an industry standard PCIe cable. (Note: the lid over the GLV in the figure is a temporary lid and not the product lid.)

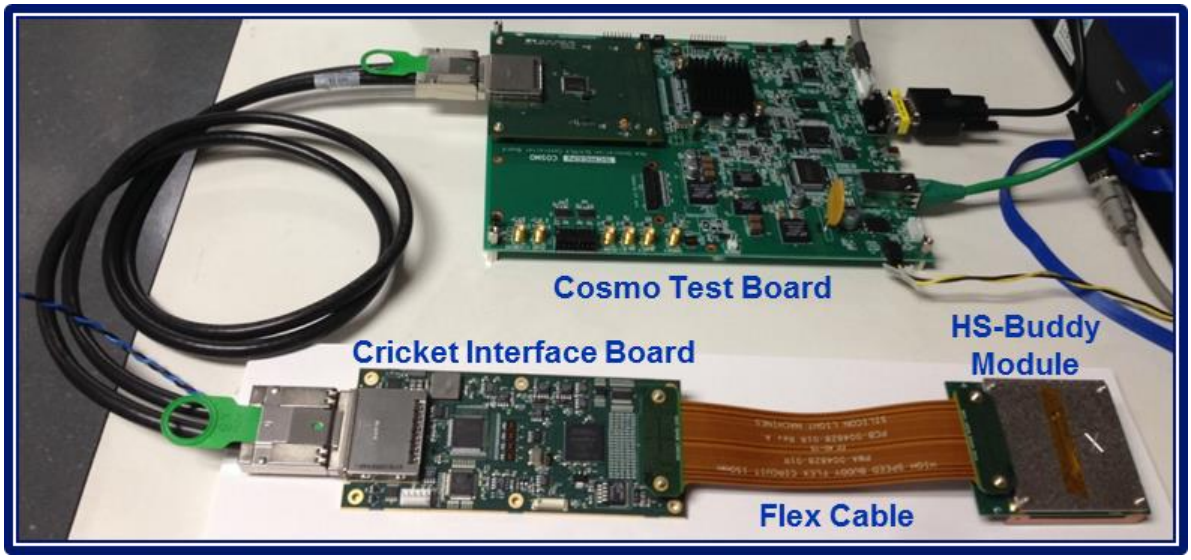

Figure 3-2 Cosmo Test Board interfaced to High-Speed Buddy & Cricket

3.1 Major Components & Functions

3.1.1 GLV Electrical Description

The electrical interface to the GLV device consists of the following:

- Active ribbons
- Bias ribbons
- Common
- Seal Ring

A GLV pixel consists of 3 active ribbons interlaced with 3 bias ribbons. The 3 active ribbons of a pixel are electrically interconnected. There are 1088 bond pads where each pad connects to a 3 active ribbon pixel set. These 1088 bond pads are driven by the 272 high voltage outputs of four SLM212 driver chips. The user controls the high voltage outputs of the driver by writing digital data to the SLM212 driver chips via the 4 LVDS data channels.

All of bias ribbons are interconnected together and are driven by a dedicated 9-bit DAC in the PSoC. Likewise, the GLV common is driven by a second dedicated 9-bit DAC in the PSoC. Both of these DACs are buffered by operational amplifiers before they drive the bias and common pads on the GLV. The user controls the 9-bit DACs by writing data to the PSoC via the I^2C interface.

The seal ring pads of the GLV are directly connected to ground on the module.

3.1.2 SLM212 Driver

The SLM212 driver chip is a design that surpasses the performance of the previous generation driver chips in a number of areas: speed, functionality, analog resolution, programmable delays, and 3.3 V operations. The SLM212 is designed for use with a linear GLV array. Typically, four SLM212 driver chips will be assembled in a multi-chip module with a single 1088-pixel linear GLV array. The driver features are intended to support three operational modes.

3.1.2.1 Principle Features

- 272-channel, high voltage digital-to-analog converters (DACs)
- Three modes of operation
  - Amplitude (AMP) Mode
    - One 10-bit register per channel drives the 10-bit DAC
  - Amplitude (AMP) with Delay Mode (1)
    - Internally stored delay value per channel can delay time of pixel update
  - Static Pulse Width Modulation (PWM) Mode – 10-bit data interface (1)
    - Internally stored amplitude & delay values selected with 1-bit pixel data
    - Three 10-bit amplitude & three 8-bit delay registers per channel
    - Designed for use in digital printing applications
- DAC design
  - Amplitude resolution of 10-bits
  - Sixteen 6-bit DACs (16 segments) connected for 1024 amplitude levels
  - Design of current DAC guarantees monotonicity between 6-bit DAC segments.
  - Each 6-bit DAC has separately programmable gain (slope)
    - Each slope (segment gain) programmable to 10-bits precision
  - Current-steering/balanced power DAC & high voltage stage reduces glitch energy
  - On-chip, programmable DAC for ECREF adjustment
- Enhanced register address scheme
  - Enables partial update of amplitude registers
  - Selectable direction for writing data to channels (up or down)
- Test features including:
  - Multiplexing of analog outputs to test pins – linearized output
  - 1st and last HV channels duplicated for test outputs (TC0, TC271)
  - Parity check for data transmission integrity verification (1)
- 3.3 V digital logic for input
- Additional on-chip DACs for reference supplies (two 10-bit current DACs) (1)
- Designed in an established 0.35 μm HV CMOS process (TSMC)

(1) These features are not implemented on the High-Speed Buddy Module

3.1.2.2 Operational Modes (Basic)

The SLM212 is primarily a 272 channel high-speed DAC designed to drive a purely capacitive load within an approximately 22V range and 10bit accuracy. The device can operate in three modes: amplitude (AMP), amplitude (AMP) with delay and pulse width modulation (PWM).

In AMP mode each channel acts as an independent 10-bit voltage DAC. All 272 channels are pre-loaded with a 10-bit code and on the column strobe (COL) all 272 channels are transitioned to the new voltage level. The transition to the new voltage can take place immediately following the COL strobe. In AMP with delay feature, the time of update for each channel can be delayed relative to the COL strobe by delay increments defined by the PWM clock.

In PWM mode the device is loaded with 1 data bit per channel per column. This 1-bit data stream is used by each channel to select one of three possible 10-bit output voltages and 8-bit delay settings. After each COL strobe the delay value is used to delay the transition to the new output level by delay increments defined by the PWM clock.

[The "PWM Mode" and the "AMP with delay" feature are not described in this manual, contact Silicon Light Machines for information on how to implement these features.]

3.1.2.3 Block Diagram of SLM212 Chip

An illustration of the approximate driver die aspect ratio is shown in **Figure 3-3**. Most of the inputs and outputs are confined to the long sides of the driver chip. Most of the data input and test functions are routed through a series of low-density bond pads located along one of the long sides of the die, referred to here as the bottom side. The other long side of the die contains all 272 channels of high voltage outputs. The outputs are configured in a higher density, double, staggered row of bond pads, which are designed for chip-to-chip wire bonding to a linear GLV array device.

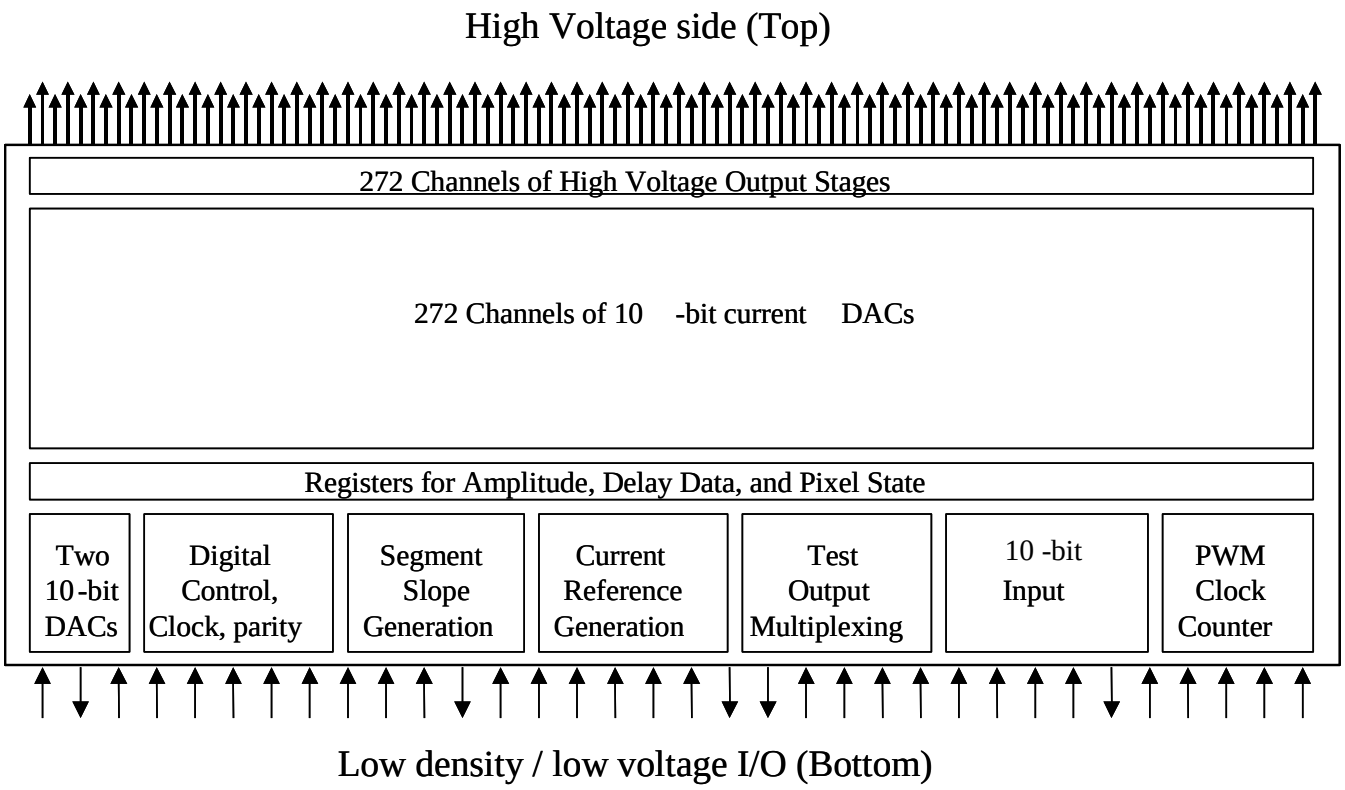

Figure 3-3 SLM212 Functional Diagram

Figure 3-3 is not an accurate layout or floor plan of the SLM212, but it is intended to illustrate a number of key aspects of the chip. That is, the data input is generally to one side of the device, and the high voltage outputs are along the other. The data is received in a serial manner (one or two channels per data clock), and is sequentially stored in registers to supply the 272 channels. Typically in AMP mode, all of the 272 high voltage outputs are updated simultaneously. Essentially, the chip accomplishes the function: "serial digital input" – to – "parallel analog output".

3.1.3 Programmable System on a Chip (PSoC)

The HS-Buddy uses a Cypress Semiconductor's Programmable System-on-Chip (PSoC) to:

- Power up (or down) voltages to the SLM212 drivers (3.3V digital, 3.3V analog and 24V);
- Adjust key voltages to the GLV via D/A conversions (High-Voltage, Bias & Common);
- Perform Built-In-Test (BIT) functions and health monitoring functions via A/D conversions;
- Provide a small amount of non-volatile memory (EEPROM) for storing GLV configuration data and initialization data as well as user-defined data.

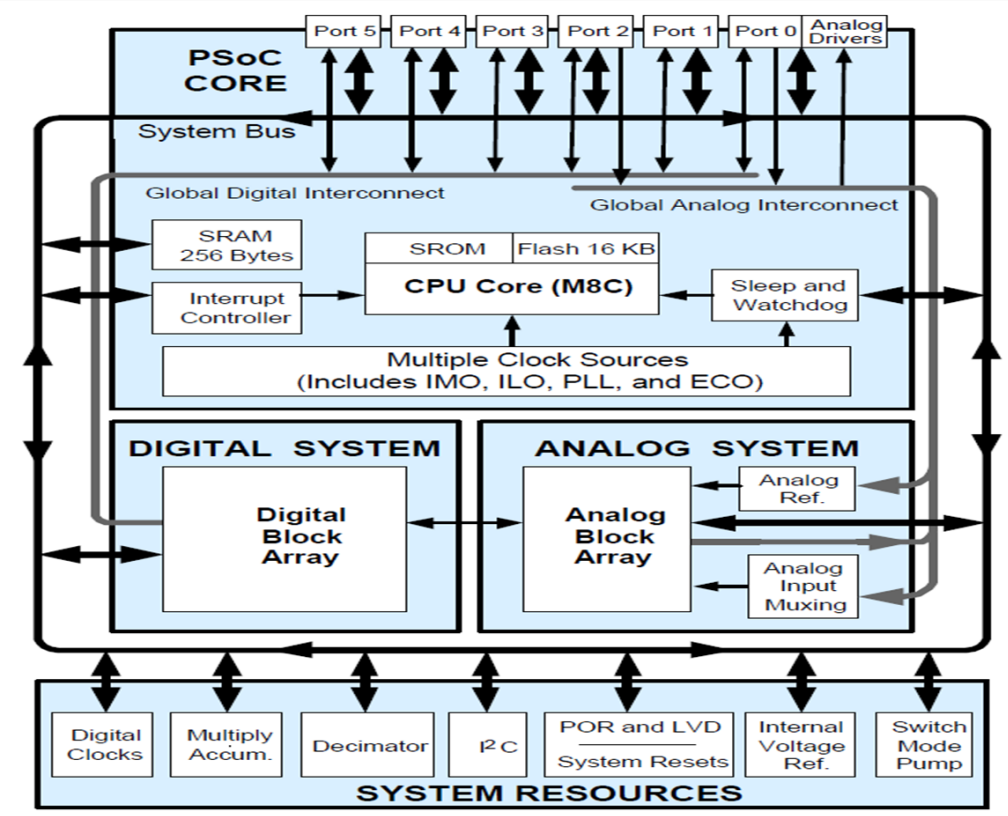

Figure 3-4 PSoC Features

(LVD=Low-Voltage-Detect, POR=Power-On-Reset)

The PSoC has a built-in 8-bit micro-controller unit, 256 bytes of SRAM, 16 Kbytes of FLASH memory, a Watchdog/Sleep timer, an interrupt controller, and low-voltage-detect (LVD) and power-on-reset (POR) circuits. It has 12 programmable analog blocks (for D/A, A/D converters, programmable gain amplifiers, analog comparators, etc.) and 8 programmable digital blocks (for timers, counters, SPI, UART, etc.). It has 40 I/O pins organized into 5 I/O ports (8 pins per port), 8 of these I/O pins can be configured as analog inputs, and 4 of them can be configured as analog inputs or outputs. The Cypress part number for the PSoC used on the High-Speed Buddy is: CY8C27543-24AXI. A block diagram of the PSoC features is illustrated in Figure 3-4.

Figure 3-5 shows the PSoC design of the High-Speed Buddy and the interfaces to other components. The PSoC is physically located on the Cricket Interface Board. The PSoC is responsible for powering up and down the SLM212 drivers by enabling or disabling the voltage regulators to the drivers. It can also enable or disable the LVDS deserializer and LVDS receiver on the Cricket Board (they are responsible for transporting pixel data to GLV via SLM212 drivers). This figure is intended on showing the interface signals to the PSoC at a functional level and is not intended to be a detailed schematic of the module. The figure does not show the downlink data path and just shows the control pins from the PSoC to the deserializer and the FPGA.

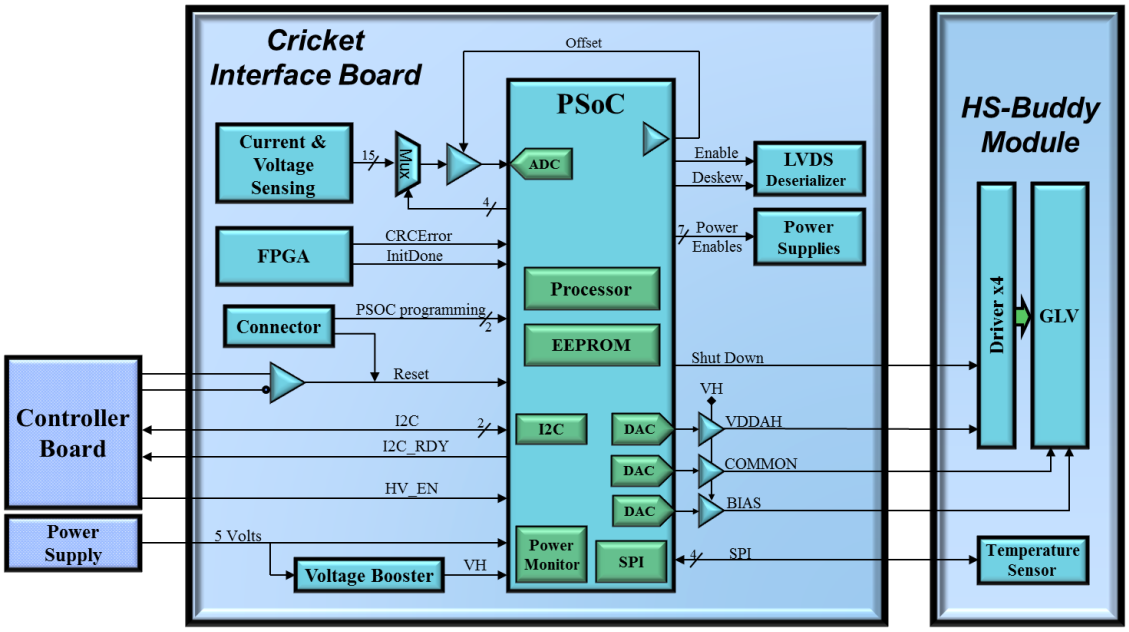

Figure 3-5 PSoC Interface Diagram

The PSoC provides three 9-bit D/A channels to control high-voltage (VDDAH), Common and Bias voltages to the GLV. It also has an 11-bit A/D converter, which with an external 16:1 multiplexer provides 16 A/D channels for monitoring on-board currents and voltages. An SPI interface allows access to a digital temperature sensor located near the GLV.

The PSoC has 4096 bytes of its internal FLASH memory as EEPROM; with 2048 bytes allocated for storing GLV initialization and configuration data and an additional 2048 bytes allocated for persistent user data. The PSoC also keeps track of the GLV module on-time (lifetime) in hours, and stores the lifetime data in the EEPROM.

The host controller (the master) may access these functions on the PSoC via an I^2C interface to the PSoC (the slave). This I^2C interface is the High-Speed Buddy module control interface. The PSoC uses a 3.3V power supply. The host controller (the master) that communicates to the PSoC via I^2C can operate at 3.3V (or 5.0V as well due to I^2C Buffer on the Cricket Board).

3.1.4 LVDS Deserializer and LVDS Receiver

The GLV pixel data is received from the external interface via eight channels of serialized data and the associated data clock. These data channels and clock use Low Voltage Differential Signaling (LVDS). The module has a deserializer integrated circuit that is manufactured by National Semiconductor (part number: DS90CR486). The deserializer translates the eight channels of high-speed serial LVDS data into 48 bits of LVTTL parallel data. The DS90CR486 also has a phase locked loop (PLL) that allows the LVDS clock channel to be transmitted at a much lower frequency than the LVDS data channels. Specifically, there are seven LVDS data bits for each LVDS clock cycle on each data channel.

When interfacing to the LVDS deserializer, a National Semiconductor LVDS serializer (part number: DS90CR485), or similar device, should be used.

The COL and WCLK input signals are received from the external interface via dedicated LVDS channels (e.g. these channels are not serialized). The LVDS receiver is a low skew, two-channel receiver and is manufactured by Fairchild Semiconductor (part number: FIN1048). When interfacing to the LVDS receiver, a low-skew LVDS transmitter should be used.

In addition, an input RESET signal is received from the external interface via a dedicated LVDS channel. When interfacing to this LVDS receiver, a LVDS transmitter should be used.

All LVDS signals are terminated differentially on the Cricket Board with 100 ohm resistors.

3.1.5 Field Programmable Gate Array (FPGA)

The FPGA provides a bridge function between the 48-bit deserializer output and the 88-bits of the 4 SLM212 drivers. The FPGA takes four lanes of 10-bit data plus 2 control bits from the deserializer output and translates the data into four lanes of 20 data bits and 2 control bits going into the inputs for the 4 SLM212 drivers. The SLM212 driver input buses run at half the speed of the deserializers output bus. In addition, a PLL in the FPGA optimizes the duty cycle of the clock and optimizes the phase of the clock to the data. A functional block diagram of one of the 4 data lanes of the FPGA is illustrated in Figure 3-6.

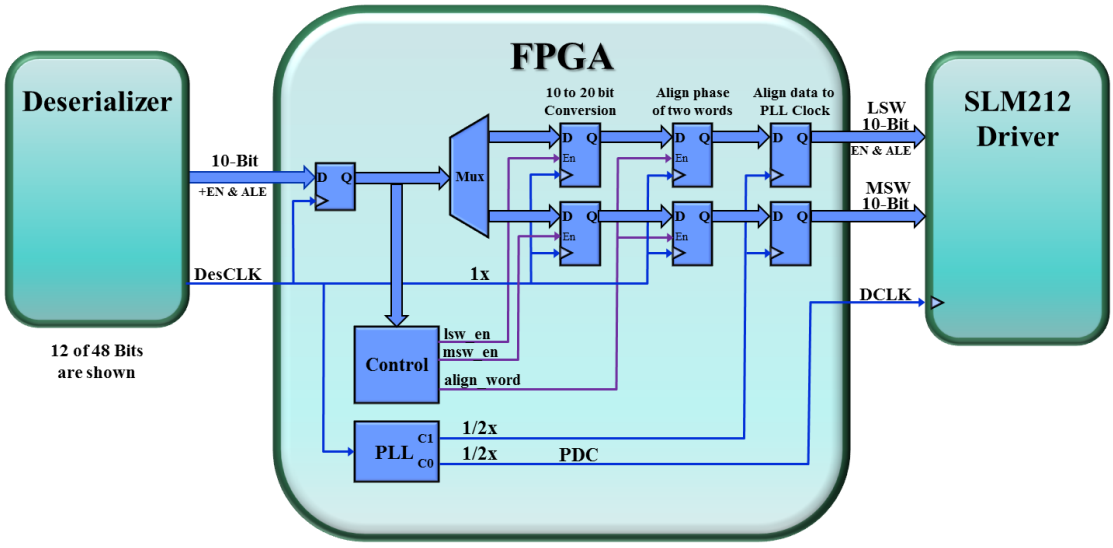

Figure 3-6 FPGA Block Diagram (1 of 4 data lanes)

3.1.6 Internal Power Supply

The internal power supply on the Cricket Interface Board generates the voltages needed to power the components on both the Cricket Board and the High-Speed Buddy Module. Figure 3-7 illustrates the internal power supply on the Cricket Interface Board. The net names highlighted in green are monitored by the PSoC health monitoring system and can be read by the user via the I^2C A/D function. The common and bias voltages for the GLV are generated by DAC's in the PSoC. In addition, the VDDAH voltage to the GLV is controlled by a third DAC in the PSoC. LDO in the figure refers to a Low Drop Out voltage regulator. The PSoC enables the power supply components during a power up sequence and shuts down the power supply components during a power down sequence. Theses sequences are described in the next section.

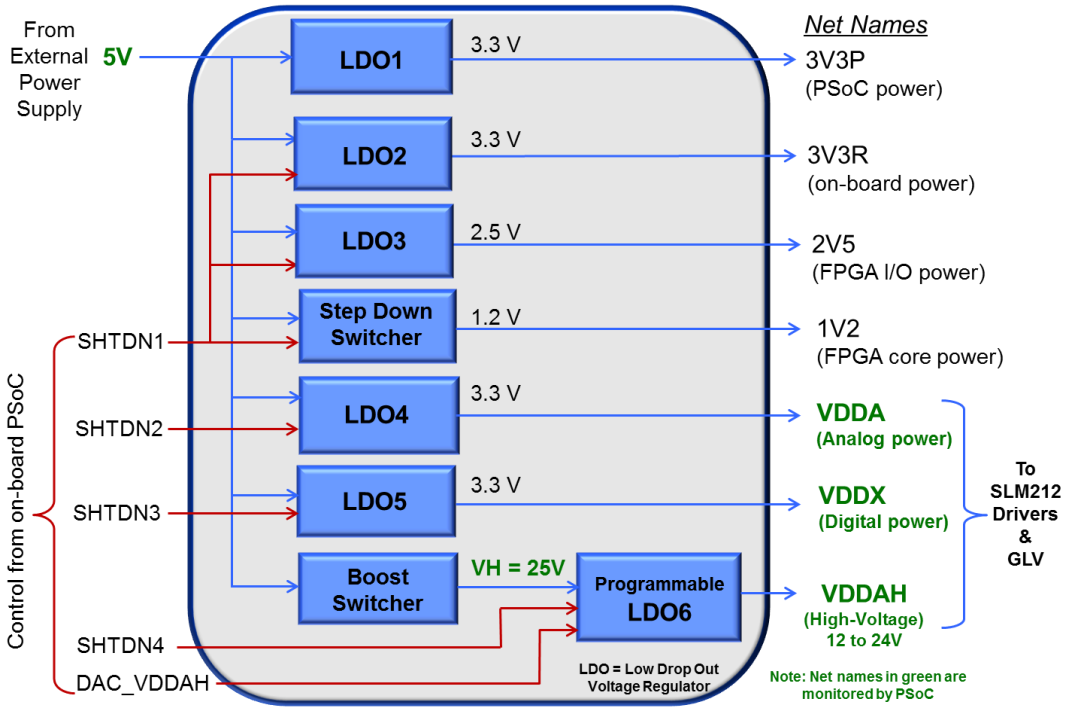

Figure 3-7 Internal Power Supply Block Diagram

3.1.7 Voltages that Drive the Pixel Ribbons

The following control voltages affect the final voltage that is applied across a pixel’s ribbons. The voltages that are required to drive a single GLV pixel are illustrated in Figure 3-8.

- Amplitude DAC Voltage
  - Each pixel is driven by a dedicated 10-bit DAC in one SLM212 driver channel
  - Each DAC output is amplified to a High Voltage Output (HVO) in the driver channel and drives the 3 active ribbons of a single pixel
- EcRef Voltage
  - Each driver has a global EcRef that affects the DAC voltage response
- Slope Voltages
  - Each driver has 16 slope segments that affect the DAC voltage response
- High Voltage Power Supply Voltage (Vddah)
  - Global power supply voltage to the driver channel’s high-voltage amplifier
- Bias Voltage
  - Global voltage that connects to all bias ribbons
- Common Voltage
  - Global voltage to the common plane (i.e. substrate below the ribbons)

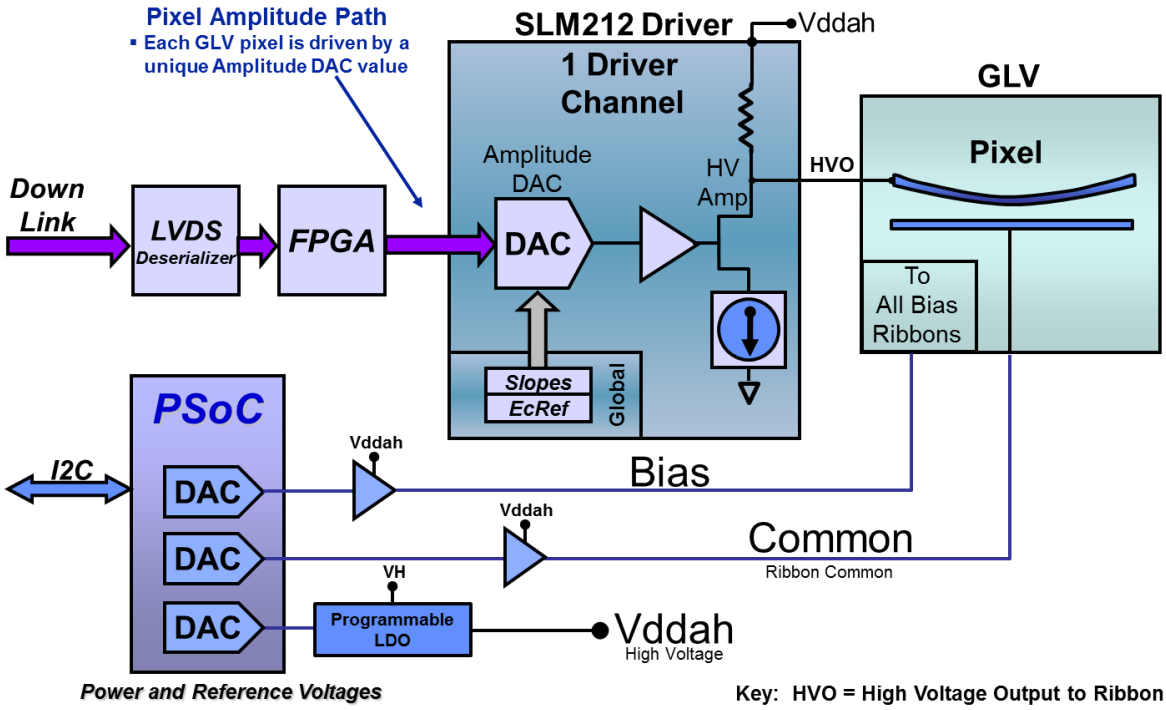

Figure 3-8 Voltages that Drive the Ribbons

The 4 SLM212 drivers provide a dedicated 10-bit amplitude DAC for each pixel in the GLV. The 10-bit amplitude DAC values are sent over the downlink interface via the deserializer and the FPGA to the SLM212 driver channel. Each SLM212 driver has a global EcRef circuit and 16 slope segments to optimize the DAC output response. The EcRef and slope values are written to the SLM212 driver via the downlink interface during initialization.

The high voltage power supply (Vddah) supplies power to the high voltage amplifier in a SLM212 driver channel. The high voltage amplifier converts the low voltage from the amplitude DAC output to the high voltage required by the MEMS ribbons. The output of the high voltage amplifier is the High Voltage Output (HVO) of the driver channel which connects to the 3 active ribbons of a pixel. The voltage across an active ribbon is the difference between the HVO voltage and the global common voltage. The Vddah voltage can be adjusted by a 9-bit DAC that is located in the PSoC.

A single bias voltage connects to all bias ribbons in the GLV. The voltage across a bias ribbon is the difference between the bias voltage and the common voltage. The bias voltage can be adjusted by a 9-bit DAC that is located in the PSoC.

The common voltage connects to the substrate electrode located under the ribbon array. The voltage across an active ribbon is the difference between the HVO voltage and the common voltage. The common voltage can be adjusted by a 9-bit DAC that is located in the PSoC.

The three 9-bit DAC's that generate the Vddah, bias and common voltages are physically located in the PSoC and have DAC ranges from 0 to 510. The PSoC DAC's are controlled by sending DAC values over the I2C interface using the D/A conversion protocol.

Warning: GLV Ribbon Over-Voltage

The GLV is a capacitive MEMS device. Like all capacitive microstructures, the device exhibits "pull-in" (also called "snap-down") when the potential between the ribbon and substrate exceeds the pull-in voltage. The pull-in voltage is a GLV property determined by the physical geometry of the ribbons and the gap, as well as the ribbon material properties (e.g. stress). In most cases, GLV ribbons survive snap-down, but in some cases the ribbon may remain permanently stuck to the substrate, rendering it useless. For this reason, the user must always be vigilant in keeping the applied voltage below the pull-in voltage.

Each module is shipped with recommended operating conditions and a global soft limit for Vddah. Software limits are set in the PSoC non-volatile EEPROM memory to prevent the user from programming an excessive voltage on the Vddah voltage. The PSoC prevents the Vddah DAC values from exceeding the Vddah high error level. The high error level is stored in Bank 8 of the PSoC's EEPROM memory. When the user sends a Vddah DAC setting that is above of the high error level, then the PSoC ignores the new setting and the DAC setting is not changed.

The Vddah DAC Error Limit is set during the outgoing ship test of the High-Speed Buddy for the Vddah voltage. However, there are situations that can affect the IV curve and the ribbon snap-down voltage in the field. For example, laser exposure can heat up and soften the ribbon which reduces the voltage required to deflect the ribbon. The amount of laser heating is dependent on the user's optical design and the power of the user's laser. The pre-stored Vddah error limit cannot account for the user's configuration. Therefore, the DAC Error Limit cannot guarantee that ribbon snap-down does not occur.

In addition, the intensity-voltage response of a GLV pixel is sensitive to even small variations in ribbon geometry or properties. For example, Figure 3-9a shows the 0th order I-V curves when the sacrificial layer thickness varies ±3%. The wavelength used for these IV curves is 808nm. The crosses denote the pull-in voltage for each device, and one notes that these can vary substantially. Typical operation requires modulating the ribbons between the bright state (zero applied voltage) and the dark rollover (white dots) where the ribbon has been deflected by one quarter wavelength. In this extreme example, one notes that if the green pixel is driven to dark rollover, the magenta pixel would be driven past its snap-down voltage. This highlights the importance of monitoring and controlling the voltage applied on a per-pixel basis. This is most directly accomplished by monitoring the individual GLV pixel I-V curves,

as they directly describe deflection. “Optical over-drive” is a good method for characterizing ribbon deflection past quarter wave. As shown in Figure 3-9b, optical overdrive is the % intensity change past the quarter wave rollover point. Generally we recommend not exceeding 25% optical overdrive, though the true limit depends strongly on operating wavelength.

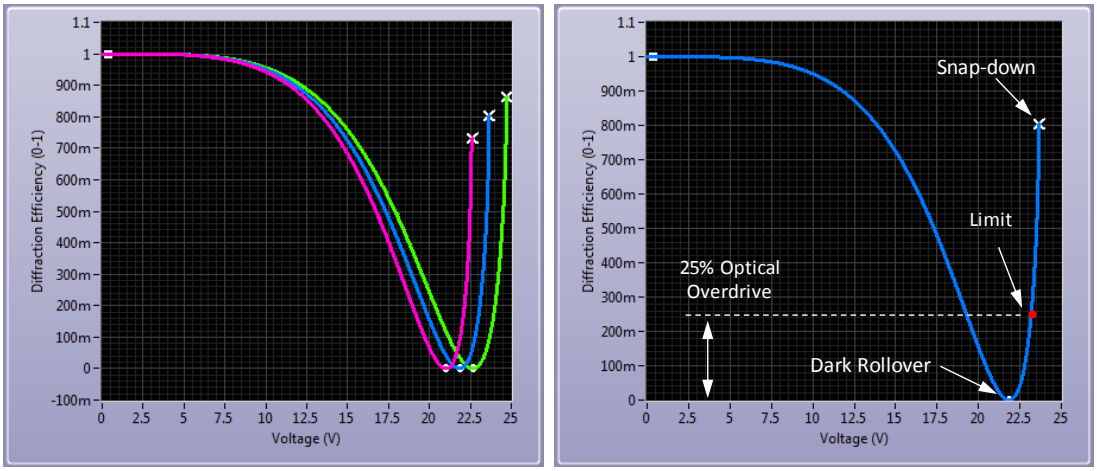

Figure 3-9 (a) I-V curves for identical GLV ribbons with ±3% variation in sacrificial gap thickness. (b) Limiting applied voltage beyond rollover by tracking “optical-overdrive” (25% overdrive limit shown here)

In summary, the GLV is equipped with global voltage protection (Vddah soft limit) to prevent ribbon over-voltage. However, this limit may not protect the GLV under all field operating conditions (e.g. high power softening). Moreover, the Vddah soft limit does not account for pixel-to-pixel variation which must also be considered to properly protect the GLV from snap-down. Thus to achieve optimum performance and snap-down protection, the user is recommended to take the following approach (preferably incorporated into an automated GLV calibration algorithm):

1. Measure individual pixel IV curves for all pixels
2. Adjust Vddah to the minimum value which achieves rollover on all pixels at DAC 1023
3. Limit amplitude DAC value on all other pixels to not exceed 25% optical overdrive.

3.2 Interfaces

The interface to the High-Speed Buddy contains the following groups of signals:

- Downlink Interface (Pixel Data)
- Control Interface (I^2C)
- Timing Interface (Column Strobe & WCLK)
- Other Signals (including control, status & test pins)
- Power Supply

Figure 3-10 illustrates the interface to the High-Speed Buddy. All signals connect from the user's Controller Board to the Cricket Interface Board. An external power supply is used to provide 5 volt power.

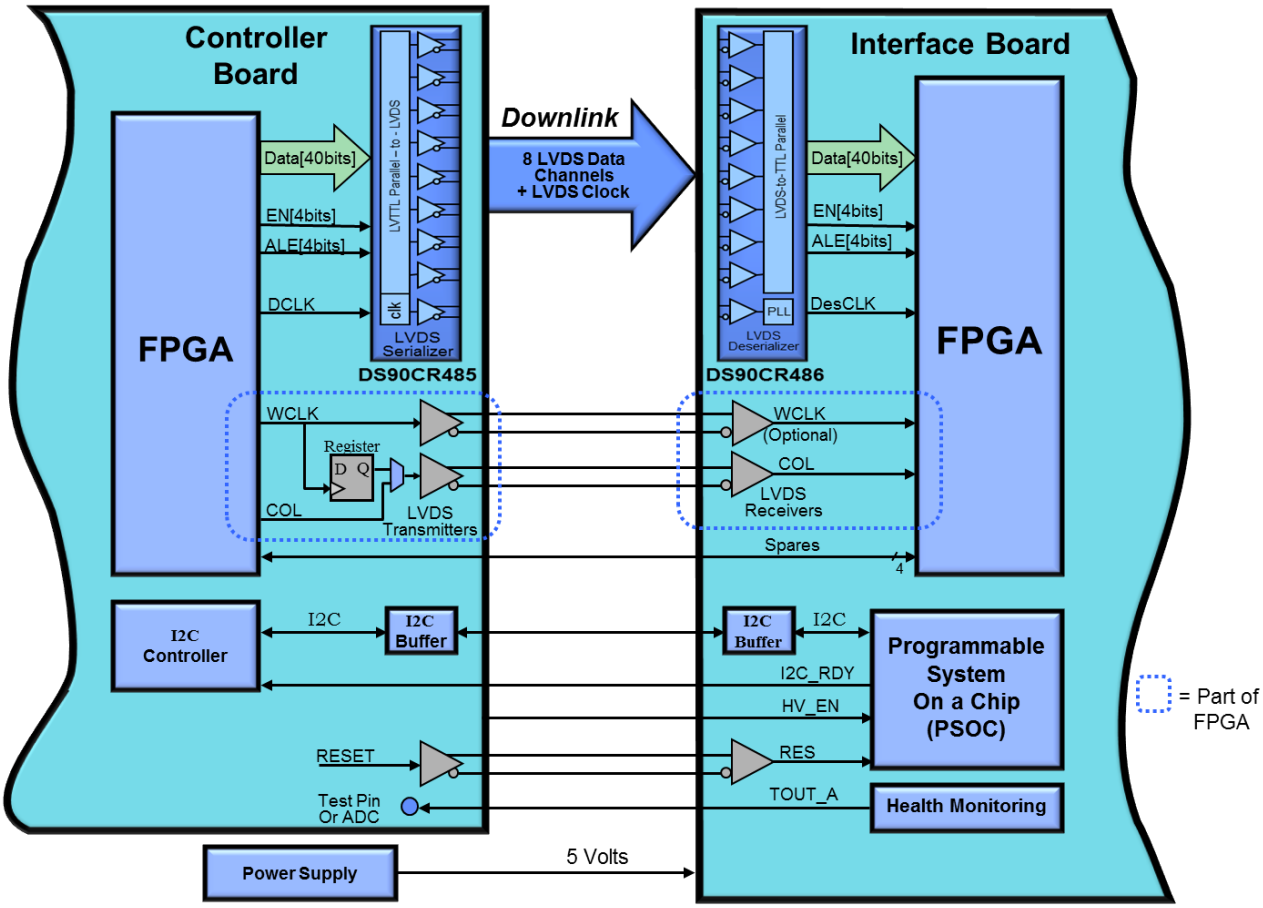

Figure 3-10 Signal Interface to the Cricket Interface Board

The user interfaces to the High-Speed Buddy using two connectors located on the Cricket Interface Board. One is a Data Connector (J1) and the second is a power connector (J3). All input and output signals are interfaced through the J1 connector. The 5 volt power is supplied to the J3 power connector. *Contact Silicon Light Machines to request a copy of sample schematics for the interface circuitry that should be implemented on a user's controller board.*

J1 Data Connector

The J1 connector is a 68-pin Molex iPASS x8 Receptacle (0.8mm pitch, right-angle, surface mount) manufactured by Molex (part number: 75586-0007). The receptacle uses an iPASS x8 EMI Guide Housing also manufactured by Molex (part number: 745400201). The same receptacle and housing are used on the user's controller board to connect to the cable.

The pins of the J1 connector are defined in **Table 3-1**. This table defines the signal names, the directions (in or out), the voltage levels, the pin assignment and the description of each signal.

	Signal	Pin # (Controller Board)	J1 Pin # (Cricket)	In/Out	Levels	Description	Notes
Downlink	RxIn7+	A33	B33	In	LVDS	Data Channels (8 Channels)	
	RxIn7-	A32	B32				
	RxIn6+	A30	B30				
	RxIn6-	A29	B29				
	RxIn5+	A27	B27				
	RxIn5-	A26	B26				
	RxIn4+	A24	B24				
	RxIn4-	A23	B23				
	RxIn3+	A12	B12				
	RxIn3-	A11	B11				
	RxIn2+	A9	B9				
	RxIn2-	A8	B8				
	RxIn1+	A6	B6				
	RxIn1-	A5	B5				
	RxIn0+	A3	B3				
	RxIn0-	A2	B2				
	RxClk+	A15	A15	In	LVDS	Data Clock	
	RxClk-	A14	A14				
Timing Interface	COL+	B3	A3	In	LVDS	Column Strobe	1
	COL-	B2	A2				
	WCLK+	B6	A6	In	LVDS	PWM Clock (Reserved for growth)	
	WCLK-	B5	A5				
Control Interface	I2C_SDA	B30	A30	In/Out	3.3V	I2C Data	2
	I2C_SCL	B33	A33	In/Out	3.3V	I2C Clock	2
Others	RES+	B27	A27	In	LVDS	Reset for PSoC	1
	RES-	B26	A26				
	HV_EN	B20	B20	In	3.3V	Enable for high voltage (VDDAH)	1
	I2C_RDY	A21	A21	Out	3.3V	PSoC is ready for an I2C command	1
	TOUT_A	B21	B21	Out	Analog	Analog Test Out from Drivers	
	SPARE3	A19	A19	I/O	3.3V	Spares to FPGA	3
	SPARE2	A20	A20	I/O	3.3V	Spares to FPGA	3
	SPARE1	B24	A24	I/O	3.3V	Spares to FPGA	3
	SPARE0	B23	A23	I/O	3.3V	Spares to FPGA	3
	SPARE_G	B29	A29			No-connect on controller board	
Ground	Ground	A34,A31,A28, A25,A22,A16,A13, A10, A7, A4, A1 B34, B32,B31,B28,B25, B22,B13,B11,B10, B8, B7, B4, B1	A34,A32,A31,A28, A25,A22,A16,A13, A10, A7, A4, A1, B34,B31, B28,B25,B22,B13, B10, B7, B4, B1		GND		
No Connect	NC	A18,A17, B19,B18,B17,B16, B15,B14,B12,B9	A18,A17 A12,A11, A9, A8, B19,B18, B17,B16,B15,B14		-		

Notes: (1) Active high.
(2) I2C signals will be pulled up to 3.3 volts on Cricket Board.
(3) Spares are connected to FPGA pin and are reserved for growth. Controller FPGA should tri-state pins.

Table 3-1 Pin Definitions for J1 Connector

An off-the-shelf iPass x8 Cable can be used to connect the Controller Board to the J1 Data Connector on the Cricket Interface Board. Figure 3-11 shows a schematic of the x8 iPass Cable. Wires intended for differential pairs make connection from the A side to the B side. Wires intended for single ended signals make connection to the same side (i.e. A connects to A and B connects to B). There are a few cases that single-ended signal were assigned to wires that are typically used for differential pairs. Table 3-1 shows pin assignments for both the J1 connector on the Cricket and the connector on the other side of the cable (on the controller board). The following off-the-shelf cables are examples of iPass x8 cables (68-wire, shielded, plug-to-plug) that can be used for connection to Cricket's J1 Data Connector. Other lengths are also available.

- Molex p/n: 0745460813 (0.5 meter) or Molex p/n: 0745460801 (1 meter) or
- Molex p/n: 0745460803 (3 meter) or Molex p/n: 0745460805 (5 meter)

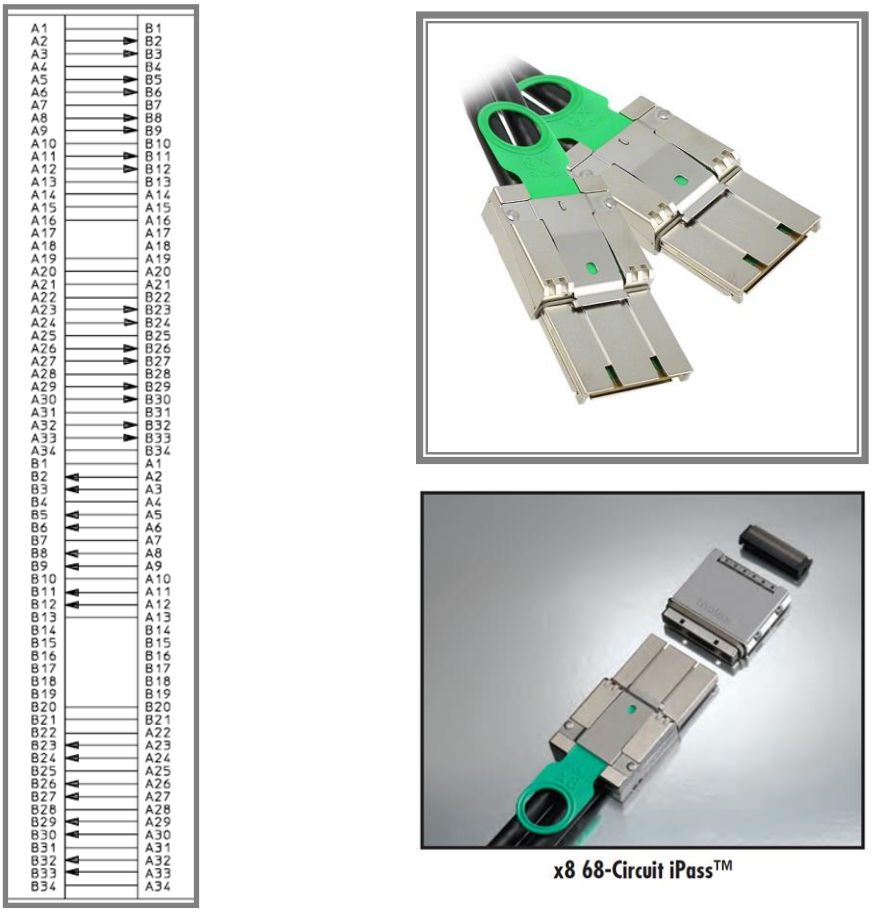

Figure 3-11 iPass x8 Cable Pin Assignments
(a) Cable schematic; (b) cable connectors, & (c) cable to housing & header

J3 Power Connector

The J3 power connector is a 2-pin Micro-Fit header (3mm pitch, right-angle, surface mount), manufactured by Molex (p/n: 43650-0213). The typical mating connector that is on the end of the power cable is Molex Micro-Fit housing (p/n: 43645-0200) with crimp terminal (p/n: 43030). Crimp tool: (p/n: 63811-2800 or 63819-0000).

J3 Pin Assignments:

- Pin 1: 5.0 volts @ 3 amps (max); 2 amps (typical)
- Pin 2: Ground

3.2.1 Downlink Interface

The primary purpose of the downlink interface is to transport the pixel data to the GLV. The downlink interface includes the eight LVDS channels (RxIn[7:0]) of serialized data that are transported from the LVDS serializer to the LVDS deserializer. The user writes downlink data to the 24-bit data input port of the serializer on the controller board. The data is serialized and transmitted over the 8 high-speed LVDS channels to the deserializer on the Cricket Interface Board. The Cricket’s FPGA translates the LVDS deserializer output data to the data inputs of the four SLM212 driver chips. The SLM212 drivers convert the digital pixel values into the high-voltage drive levels that are required to displace the GLV ribbons.

The SLM212 driver data interface is defined in Appendix B “Driver Data Interface Description”. To understand the SLM212 driver data interface, the user must understand the SLM212 driver interface. Therefore, the information presented in Appendix B should be read prior to reading this section.

LVDS Signals

Although the LVDS signals are connected to the J1 connector pins, the interface will be defined at the input to the LVDS serializer on the user’s controller board. This is necessary because the LVDS signals have been serialized in a pattern that is good for transmission, but rather poor for describing a logical interface. However, the electrical interface between the LVDS transmitter output on the users board and the input to the LVDS deserializer on the module is a well-defined interface and documented in the device manufacturer’s data sheet (DS90CR486/DS90CR485; National Semiconductor).

When using the DS90CR485 LVDS Serializer (or equivalent), the user only needs to connect the 8 differential pairs from the LVDS Serializer to the appropriate pins on the J1 connector of the module. However, providing a high quality data transmission path for the LVDS differential pairs is essential. The skew between the two signals of the differential pairs should be minimal. The path should be a controlled impedance path with differential impedance of 100 ohms. The iPass x8 cable assembly from Molex meets the electrical requirements of the LVDS data path.

Downlink Data Path

The downlink is used to write to the four SLM212 drivers on the module. Each SLM212 driver operates with a dedicated 20-bit data bus and two control signals (enable “EN” and address latch enable “ALE”). The 20-bit bus allows the user to write two 10-bit GLV pixel amplitude values (pixel pair) in a single SLM212 driver bus cycle. With 4 SLM212 driver chips, there are a total of 80 data bits plus 8 control bits (88-bits). The downlink data path is: the serializer (24-bit bus DDR input), deserializer (48-bit bus output), the FPGA and then the SLM212 driver inputs. The DS90CR485 serializer has a 24-bit double data rate (DDR) bus which generates a 48-bit word by combining two 24-bit words. The FPGA translated the 48-bit deserializer output into the 88-bit inputs for the four SLM212 drivers. The serializer, deserializer and the FPGA create a bridge from the serializer input to the SLM212 driver input data bus. The downlink data path is illustrated in Figure 3-12.

This figure also shows the numbering and orientation of both the GLV pixels and the SLM212 driver channels. The 1088 GLV pixels are number from 1 to 1088. However, the 272 SLM212 driver channels are numbered from 0 to 271. Drivers U1 & U2 are physically oriented so that the lowest driver channel (# 0) is connected to the lowest numbered GLV pixel. These driver channel pairs are sequentially loaded from lowest channel to highest, and the driver’s channel address counter is counting up. Drivers U1 & U2 are physically rotated so the highest driver channel is connected to the lowest numbered pixel. These driver channel pairs are sequentially loaded from

highest channel to lowest, and the channel address counter is counting down. Therefore, the first loaded pixel pairs are: U3 & U4: channels 0 & 1 and U1 & U2: channels 270 and 271.

The four 20-bit buses to the four SLM212 drivers are DA, DB, DC and DD. The reference designators of the 4 SLM212 driver chip are: U1, U2, U3 and U4. The list below shows each data bus, the SLM212 driver and the associated pixels of the GLV.

- DA[19:0]; U1; Odd Pixels from 1 to 543
- DB[19:0]; U3; Even Pixels from 2 to 544
- DC[19:0]; U2; Odd Pixels from 545 to 1087
- DD[19:0]; U4; Even Pixels from 546 to 1088

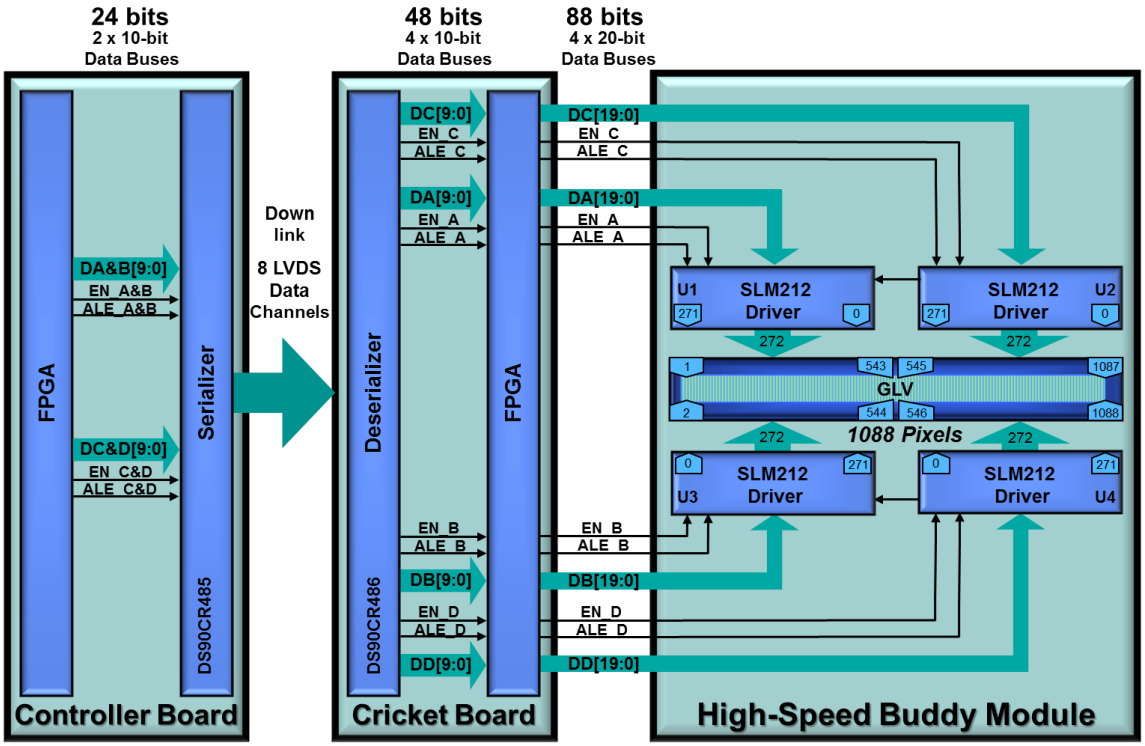

Figure 3-12 Downlink Data Path

Table 3-2 shows the data bit assignments for each component in the downlink. The user writes the driver data to the serializer in a 24-bit format. The 24-bits of the serializer input are mapped to the GLV pixels as described below and in Table 3-2. Note that in the table, E1 is the falling edge of the serializer clock and E2 is the rising edge. Two 24-bit serializer words are combined into a 48-bit deserializer output word by first sampling the E1 (falling edge) word and the following E2 (rising edge) word (see Figure 3-13). Four independent EN signals allow the user to select which drivers are written to on each bus cycle.

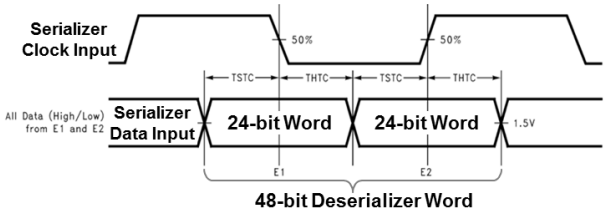

Figure 3-13 Combining two 24-bit serializer words into one 48-bit deserializer word

- Data written to Serializer input data pins on FALLING edge of clock:
  - Serializer D0-D9 will go to SLM212 U1 & drive ODD GLV pixels from 1 to 543
  - Serializer D12-D21 will go to SLM212 U2 & drive ODD GLV pixels from 545 to 1087
- Data written to Serializer input data pins on RISING edge of clock:
  - Serializer D0-D9 will go to SLM212 U3 & drive EVEN GLV pixels from 2 to 544
  - Serializer D12-D21 will go to SLM212 U4 & drive EVEN GLV pixels from 546 to 1088

DS90CR485 Serializer Input	DS90CR486 Deserializer Output	SLM212 Input	SLM212 Reference Locator	GLV Pixel Numbers
D0-D9 (E2↑)	RxOut [0-9]	DB[0-19]	U3	2 TO 544 (EVENS)
D10 (E2↑)	RxOut 10	EN_B		
D11 (E2↑)	RxOut 11	ALE_B		
D12-D21 (E2↑)	RxOut[12-21]	DD[0-19]	U4	546 TO 1088 (EVENS)
D22 (E2↑)	RxOut 22	EN_D		
D23 (E2↑)	RxOut 23	ALE_D		
D0-D9 (E1↓)	RxOut[24-33]	DA[0-19]	U1	1 TO 543 (ODDS)
D10 (E1↓)	RxOut 34	EN_A		
D11 (E1↓)	RxOut 35	ALE_A		
D12-D21 (E1↓)	RxOut[36-45]	DC[0-19]	U2	545 TO 1087 (ODDS)
D22 (E1↓)	RxOut46	EN_C		
D23 (E1↓)	RxOut47	ALE_C		

Table 3-2 Downlink Data Bits Assignments

As the pixel data travels through components of the downlink, the bus width become wider and the data data rates become slower.

Table 3-3 shows the widths of the data buses and the associated data rates of the major components in the downlink. The serializer input data bus is half of the width but twice the data rate of the deserializer output bus. The deserializer data bus is half the width and twice the data rate as the combined four SLM212 drivers’ bus width.

The clock rates and data rates shown in the table are the maximum data rates of the High-Speed Buddy. Operating the serializer at a clock rate of 108 MHz provides for a sufficient data rate to support a column rate of 350 kHz. In many applications, it may be desirable to vary the column rate. For example, the speed of the stage in a lithographic system may vary; therefore, the column rate may need to vary accordingly. However, the clock frequency of the serializer could remain at a fixed clock rate. This fixed clock rate would need to be set to accommodate the fastest possible rate of the varying column rate.

Item	Units	DS90CR485 Serializer Input	DS90CR486 Deserializer Output	SLM212 (x4 chips) Input
Bus Width	Bits	24	48	88
Control Bits	Bits	4	8	8
Data Bits	Bits	20	40	80
Bus Width	Pixels	2	4	8
Clock (1)	MHz	108	108	54
DDR		Yes	No	No
Data Rate (1)	Mbps	216	108	54
Realtive Speed		4	2	1

Table 3-3 Data Buses & Data Rates of the Downlink Components

Amplitude 0 Register Array - Write Sequence

Writing pixel amplitude values is accomplished by writing to the SLM212 driver's Amplitude #0 Register array via the serializer data input port. Writing amplitude values to all 1088 GLV pixels requires 279 clock cycles on the serializer. The first two clock cycles are used to select the zip code of the Amplitude # 0 Register array on the four drivers. This is followed by wait states. Wait states are required prior to writing to channel pairs (0,1) and (270,271). Subsequently the pixel amplitude values can be written. Data is written on both the falling edge and the rising edges of the clock, so that 2 pixel amplitude values are written (to U1 & U2) on the falling edge and 2 pixel amplitude values are written (to U3 & U4) on the rising edge of the clock.

The 279 clock cycle sequence is listed below and is illustrated in detail in Table 3-4. The table also shows the mapping of the pixel amplitude bits to the serializer data bits. The table shows how to map the data sequence to each GLV pixel. Also, the bit locations and values for the enable (EN) and address latch enable (ALE) signals are show. ALE is asserted when a zip code is on the data bus.

1. Selects the zip code of Amplitude Register #0
2. Repeats the zip code (dummy cycle required for 10-bit to 20-bit conversion)
3. Wait state
4. Wait state
5. Write amplitude values to pixels 1, 2, 545 & 546
6. Write amplitude values to pixels 3, 4, 546 & 547
7. Write amplitude values to pixels 5, 6, 548 & 549

-
-

274. Write amplitude values to pixels 539, 540, 1083 & 1084
275. Wait state
276. Wait state
277. Write amplitude values to pixels 541, 542, 1085 & 1086
278. Write amplitude values to pixels 543, 544, 1087 & 1088

Prior to writing to an array of driver registers, both the Control register and Start Address register must be configured. This write sequence assumes that the SLM212 control registers and start address registers have been configure, as listed below.

- U1 & U2: Control register is set to count down & Start Address register is set to 270
- U3 & U4: Control register is set to count up & Start Address register is set to 0

The entire initialization sequence for the SLM212 driver is described in the next section, and includes the sequence for setting the Control register and the Start Address register.

Pixels→		545 to 1088				1 to 544			
		Serializer Input: D23-D12 drives SLM212: U2 & U4				Serializer Input: D11-D0 drives SLM 212: U1 & U3			
Clock Cycle	Clock Edge	D23 ALE U2&U4	D22 ENA U2&U4	D21-D12 Data Bus U2&U4	Destination of D23-D12	D11 ALE U1&U3	D10 ENA U1&U3	D9-D0 Data Bus U1&U3	Destination of D11-D0
1	E1↓	1	1	ZIP = 0	U2 (DC)	1	1	ZIP = 0	U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
2	E1↓				U2 (DC)				U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
3	E1↓	0	0	wait	U2 (DC)	0	0	wait	U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
4	E1↓				U2 (DC)				U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
5	E1↓	0	1	Pix 545	U2 (DC)	0	1	Pix 1	U1 (DA)
	E2↑	0	1	Pix 546	U4 (DD)	0	1	Pix 2	U3 (DB)
6	E1↓	0	1	Pix 547	U2 (DC)	0	1	Pix 3	U1 (DA)
	E2↑	0	1	Pix 548	U4 (DD)	0	1	Pix 4	U3 (DB)
7	E1↓	0	1	Pix 549	U2 (DC)	0	1	Pix 5	U1 (DA)
	E2↑	0	1	Pix 550	U4 (DD)	0	1	Pix 6	U3 (DB)
8	E1↓	0	1	Pix 551	U2 (DC)	0	1	Pix 7	U1 (DA)
	E2↑	0	1	Pix 552	U4 (DD)	0	1	Pix 8	U3 (DB)
•		•	•	•	•	•	•	•	•
•		•	•	•	•	•	•	•	•
273	E1↓	0	1	Pix 1081	U2 (DC)	0	1	Pix 537	U1 (DA)
	E2↑	0	1	Pix 1082	U4 (DD)	0	1	Pix 538	U3 (DB)
274	E1↓	0	1	Pix 1083	U2 (DC)	0	1	Pix 539	U1 (DA)
	E2↑	0	1	Pix 1084	U4 (DD)	0	1	Pix 540	U3 (DB)
275	E1↓	0	0	wait	U2 (DC)	0	0	wait	U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
276	E1↓				U2 (DC)				U1 (DA)
	E2↑				U4 (DD)				U3 (DB)
277	E1↓	0	1	Pix 1085	U2 (DC)	0	1	Pix 541	U1 (DA)
	E2↑	0	1	Pix 1086	U4 (DD)	0	1	Pix 542	U3 (DB)
278	E1↓	0	1	Pix 1087	U2 (DC)	0	1	Pix 543	U1 (DA)
	E2↑	0	1	Pix 1088	U4 (DD)	0	1	Pix 544	U3 (DB)
279	E1↓	0	0	idle	U2 (DC)	0	0	idle	U1 (DA)
	E2↑	0	0	idle	U4 (DD)	0	0	idle	U3 (DB)

Table 3-4 Sequence to Write Pixel Amplitude Values to the Amplitude 0 Array of Registers

Initialization of the SLM212 Drivers

Initializing the SLM212 drivers requires that the driver control registers are set to the appropriate values. The control registers are single 10-bit word registers plus the array of 16 slope registers. Writing to a single register in the SLM212 driver is fairly straightforward and has been adequately explained in Appendix B. When writing to a driver register on a module level, the user must map the 10-bit data onto the l 10-bits of the serializer data input port and also enable the appropriate control signals (ALE & EN). This mapping has been defined in Table 3-2. For example, when writing to a single register in SLM212 driver U4, the 10-bit word is placed on the serializer data bus D[23:12] and the U4 control signals ALE and EN are mapped to serializer data bit D23 and D22, respectively.

To initialize the SLM212 driver, the user must write to the SLM212 driver registers in the sequence listed below:

1. Control Register (configure for writing to slope register)
2. Start Address Register (configure for writing to slope registers)
3. Slope Array Registers
4. EcRef Register
5. DAC0 Register (Optional)
6. DAC1 Register (Optional)
7. Analog Test Register
8. Data Transfer Register
9. Digital Test Register
10. Control Register (configure for writing to slope registers)
11. Start Address Register (configure for writing to slope registers)

Table 3-5 and Table 3-6 illustrate the 11 step sequence to initialize the driver registers with recommended default values. Only the EcRef and slope registers values need to change relative to the table. The actual values written to the slope and EcRef registers may vary from GLV module to GLV module. The slope and EcRef values are selected to optimize the IV curve of the GLV. However, good default values for the EcRef and Slope registers are:

- EcRef = 512
- Slopes[15:0] = 64

The SLM212 driver supports a 20-bit data bus to allow writing two pixel amplitude values in one clock cycle. The full 20-bit bus is supported only for writing to the driver's Amplitude #0 Register array. However, for all other SLM212 driver registers, only a 10-bit bus is supported, while the other 10-bits are ignored. Since the other 10-bits are ignored, it is easiest to just duplicate the 10-bit register data value on both 10-bit segments of the 20-bit bus. Similarly, the zip code can be duplicated on the second 10-bit segment of the 20-bit bus. The SLM212 initialization requirements can be met if the sequence documented in the initialization table is followed.

DAC0 and DAC1 are extra DAC's in the SLM212 driver that were intended for arbitrary functions. The DAC0 and DAC1 outputs are not used on the High-Speed Buddy Module, so it is not required to write values to the DAC0 and DAC1 registers. They are shown in the initialization sequence so that these registers are put into a known condition.

Notes for the SLM212 initialization write sequence tables:

- Any value with a suffix of "h" indicates a 10-bit hexadecimal value
- Zip codes can be decode by reading section 5.2 [Registers and the Address Map]
- SLM212 register values shown can be decoded by reading Section 5.4 [Data Register Formats]

	Serializer Input: D23-D12 drives SLM212: U2 & U4						Serializer Input: D11-D0 drives SLM 212: U1 & U3						
Step	**Clock Cycle**	**Clock Edge**	**D23 ALE U2&U4**	**D22 ENA U2&U4**	**D21-D12 Data Bus U2&U4**	**Destination of D23-D12**	**Clock Cycle**	**Clock Edge**	**D11 ALE U1&U3**	**D10 ENA U1&U3**	**D9-D0 Data Bus U1&U3**	**Destination of D11-D0**	**SLM212 Register**
1	1	E1↓	1	1	Zip=207h	U2 (DC)	1	E1↓	1	1	Zip=207h	U1 (DA)	Control Register (for loading Slopes)
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	301h	U2 (DC)	3	E1↓	0	1	301h	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
2	1	E1↓	1	1	Zip=204h	U2 (DC)	1	E1↓	1	1	Zip=204h	U1 (DA)	Start Address (for loading Slopes)
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	0	U2 (DC)	3	E1↓	0	1	0	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
3	1	E1↓	1	1	Zip=007h	U2 (DC)	1	E1↓	1	1	Zip=007h	U1 (DA)	Slope Registers
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	Slope 0	U2 (DC)	3	E1↓	0	1	Slope 0	U1 (DA)	
		E2↑			Slope 0	U4 (DD)		E2↑			Slope 0	U3 (DB)	
	4	E1↓			Slope 0	U2 (DC)	4	E1↓			Slope 0	U1 (DA)	
		E2↑			Slope 0	U4 (DD)		E2↑			Slope 0	U3 (DB)	
	5	E1↓	0	1	Slope 1	U2 (DC)	5	E1↓	0	1	Slope 1	U1 (DA)	
		E2↑			Slope 1	U4 (DD)		E2↑			Slope 1	U3 (DB)	
	6	E1↓			Slope 1	U2 (DC)	6	E1↓			Slope 1	U1 (DA)	
		E2↑			Slope 1	U4 (DD)		E2↑			Slope 1	U3 (DB)	
	7	E1↓	0	1	Slope 2	U2 (DC)	7	E1↓	0	1	Slope 2	U1 (DA)	
		E2↑			Slope 2	U4 (DD)		E2↑			Slope 2	U3 (DB)	
	8	E1↓			Slope 2	U2 (DC)	8	E1↓			Slope 2	U1 (DA)	
		E2↑			Slope 2	U4 (DD)		E2↑			Slope 2	U3 (DB)	
	…	…	…	…	…	…	…	…	…	…	…	…	
	29	E1↓	0	1	Slope 13	U2 (DC)	29	E1↓	0	1	Slope 13	U1 (DA)	
		E2↑			Slope 13	U4 (DD)		E2↑			Slope 13	U3 (DB)	
	30	E1↓			Slope 13	U2 (DC)	30	E1↓			Slope 13	U1 (DA)	
		E2↑			Slope 13	U4 (DD)		E2↑			Slope 13	U3 (DB)	
	31	E1↓	0	1	Slope 14	U2 (DC)	31	E1↓	0	1	Slope 14	U1 (DA)	
		E2↑			Slope 14	U4 (DD)		E2↑			Slope 14	U3 (DB)	
	32	E1↓			Slope 14	U2 (DC)	32	E1↓			Slope 14	U1 (DA)	
		E2↑			Slope 14	U4 (DD)		E2↑			Slope 14	U3 (DB)	
	33	E1↓	0	1	Slope 15	U2 (DC)	33	E1↓	0	1	Slope 15	U1 (DA)	
		E2↑			Slope 15	U4 (DD)		E2↑			Slope 15	U3 (DB)	
	34	E1↓			Slope 15	U2 (DC)	34	E1↓			Slope 15	U1 (DA)	
		E2↑			Slope 15	U4 (DD)		E2↑			Slope 15	U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
4	1	E1↓	1	1	Zip=200h	U2 (DC)	1	E1↓	1	1	Zip=200h	U1 (DA)	ECRef
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	EcRef	U2 (DC)	3	E1↓	0	1	EcRef	U1 (DA)	
		E2↑			EcRef	U4 (DD)		E2↑			EcRef	U3 (DB)	
	4	E1↓			EcRef	U2 (DC)	4	E1↓			EcRef	U1 (DA)	
		E2↑			EcRef	U4 (DD)		E2↑			EcRef	U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…

Table 3-5 Initalization Write Sequence at Serializer Input (Steps 1-4)

	Serializer Input: D23-D12 drives SLM212: U2 & U4						Serializer Input: D11-D0 drives SLM 212: U1 & U3						
Step	**Clock Cycle**	**Clock Edge**	**D23 ALE U2&U4**	**D22 ENA U2&U4**	**D21-D12 Data Bus U2&U4**	**Destination of D23-D12**	**Clock Cycle**	**Clock Edge**	**D11 ALE U1&U3**	**D10 ENA U1&U3**	**D9-D0 Data Bus U1&U3**	**Destination of D11-D0**	**SLM212 Register**
5	1	E1↓	1	1	Zip=201h	U2 (DC)	1	E1↓	1	1	Zip=201h	U1 (DA)	DAC0 Current
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	200h	U2 (DC)	3	E1↓	0	1	200h	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
6	1	E1↓	1	1	Zip=202h	U2 (DC)	1	E1↓	1	1	Zip=202h	U1 (DA)	DAC1 Current
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	200h	U2 (DC)	3	E1↓	0	1	200h	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
7	1	E1↓	1	1	Zip=203h	U2 (DC)	1	E1↓	1	1	Zip=203h	U1 (DA)	Analog Test
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	0	U2 (DC)	3	E1↓	0	1	0	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
8	1	E1↓	1	1	Zip=305h	U2 (DC)	1	E1↓	1	1	Zip=305h	U1 (DA)	Data Transfer
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	003h	U2 (DC)	3	E1↓	0	1	003h	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
9	1	E1↓	1	1	Zip=306h	U2 (DC)	1	E1↓	1	1	Zip=306h	U1 (DA)	Digital Test
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	0	U2 (DC)	3	E1↓	0	1	0	U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	4	E1↓				U2 (DC)	4	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
10	1	E1↓	1	1	Zip=207h	U2 (DC)	1	E1↓	1	1	Zip=207h	U1 (DA)	Control Register (for loading Amp0)
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	300h	U2 (DC)	3	E1↓	0	1	300h	U1 (DA)	
		E2↑			301h	U4 (DD)		E2↑			301h	U3 (DB)	
	4	E1↓			300h	U2 (DC)	4	E1↓			300h	U1 (DA)	
		E2↑			301h	U4 (DD)		E2↑			301h	U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
11	1	E1↓	1	1	Zip=204h	U2 (DC)	1	E1↓	1	1	Zip=204h	U1 (DA)	Start Address (for loading Amp0)
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	2	E1↓				U2 (DC)	2	E1↓				U1 (DA)	
		E2↑				U4 (DD)		E2↑				U3 (DB)	
	3	E1↓	0	1	270	U2 (DC)	3	E1↓	0	1	270	U1 (DA)	
		E2↑			0	U4 (DD)		E2↑			0	U3 (DB)	
	4	E1↓			270	U2 (DC)	4	E1↓			270	U1 (DA)	
		E2↑			0	U4 (DD)		E2↑			0	U3 (DB)	
…	…	…	0	0	IDLE	…	…	…	0	0	IDLE	…	…
SLM212 drivers are now ready to receive pixel data by writing to amplitude 0 register array.													

Table 3-6 Initialization Write Sequence at Serializer Input (Steps 5-11)

3.2.2 Timing Interface

Two signals on the J1 I/O connector are provided for timing control. The signals are the column strobe (COL) and the PWM clock (WCLK). These signals come onto the module as LVDS differential pairs. Both of these signals are received by LVDS receivers, then buffered in the Cricket FPGA and then drive the COL and WCLK inputs of the four SLM212 driver chips. In general, COL signal causes the newly written pixel data to drive the DAC channels and the GLV pixels. The WCLK causes the PWM counter to count and works as the time reference for delaying the pixel update time. However, the precise function of the COL signal and the WCLK is dependent on the mode of operation.

In AMP Mode, the COL signal operates as an asynchronous strobe which causes all the DAC channels to be updated with the contents of the in-channel Amplitude 0 Register. In AMP Mode, the PWM counter is non-operational. Therefore, the WCLK is not used, so the user is not required to supply a clock on the WCLK input. It can be held low or high.

The user is responsible for insuring that all pixel data transmission is completed prior to asserting column strobe. Typically, the COL strobe would be asserted after the last data word in the downlink packet has been written and prior to writing the first data word for the next column. This statement is true at the input to the DAC's inside the SLM212 driver. However, the other components in the downlink make the propagation delay of the downlink packet much longer than the propagation delay of the COL strobe. The total propagation delay of the downlink packet is the sum of the following delays: serializer, deserializer, FPGA and SLM212 driver input registers. The propagation delays of the downlink packet are shown in Table 3-7.

Device	Propagation Delay Min	Propagation Delay Max
Serializer	6 x TCIP	8 x TCIP
Cable (3 meter)	15ns	15ns
Deserializer	2 x TCIP + 5ns	2 x TCIP + 15ns
FPGA	5 x TCIP + 2ns	6 x TCIP + 6ns
SLM212 Driver	6 x TCIP	8 x TCIP
Total	**19 x TCIP + 22ns**	**24 x TCIP + 36ns**

Table 3-7 Propagation Delays of Downlink Packet

The latency of the COL signal is the sum of the propagation delays in the COL path which includes the LVDS receiver on the Cricket Board plus the buffer in the Cricket FPGA. The propagation delay of the Column Strobe is shown in Table 3-8. The propagation delay of the downlink packet relative to the column strobe is shown in Table 3-9.

Device	Delay (Min)	Delay (Max)
Cable (3 meter)	15ns	15ns
LVDS Receiver (FIN1048)	1 ns	3 ns
FPGA	2 ns	7 ns
Total	**18 ns**	**25 ns**

Table 3-8 Propagation Delays of Column Strobe

Device	Delay (Min)	Delay (Max)
Column Strobe	18 ns	25 ns
Data Packet	19 x TCIP + 22ns	24 x TCIP + 36ns
Packet to COL (1)	**19 x TCIP - 3ns**	**24 x TCIP + 18ns**

Table 3-9 Propagation of Downlink Packet relative to Column Strobe

Since the downlink data path has a greater latency than the latency for COL strobe signal, the first data word for the next column could actually be written to the serializer prior to the assertion of the COL strobe on the user’s board. This allows the module to operate at a slightly higher column rate and is referred to as “advance timing”. In this case, the shorter latency of the COL signal would result in the COL signal arriving before the first data word for the next column arrives at the first driver channel.

However, a simple timing model allows the user to pulse the COL strobe and subsequently send the next data packet to the serializer. Although, this has some wasted time, the timing is easier to implement. Both simple timing and advanced timing models are illustrated in Figure 3-14.

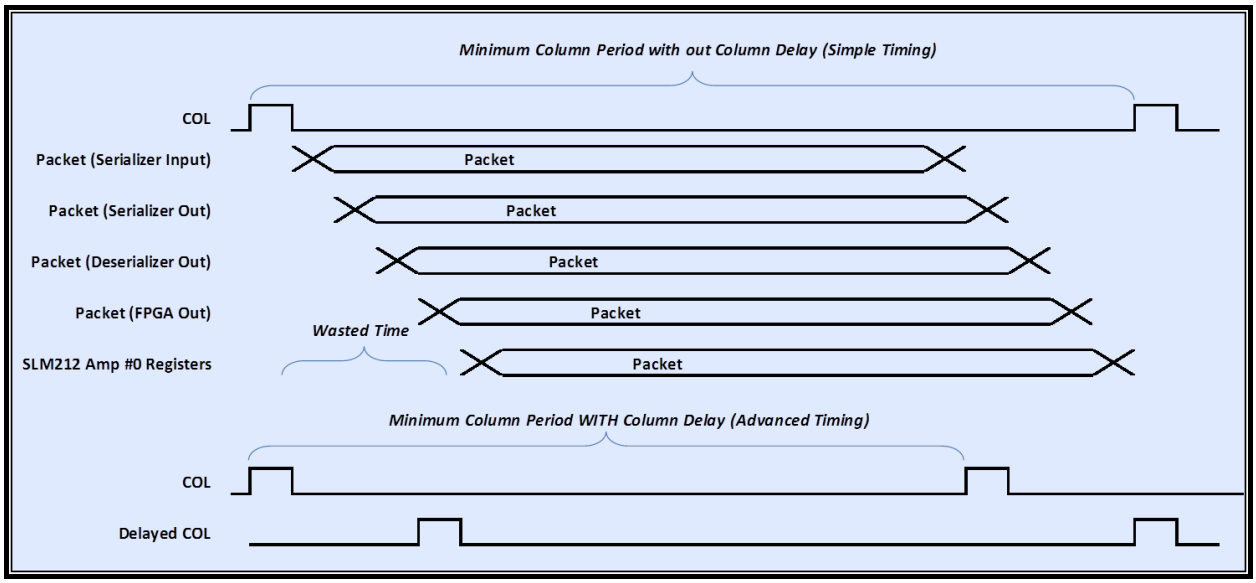

Figure 3-14 Column to Packet Timing
Simple Timing vs. Advanced Timing

The maximum serializer clock frequency that can be supported is 108 MHz. This clock frequency in conjunction with the column to packet timing requirements can result in a maximum column rate of 372 kHz when using the advance timing model. Maximum column rates of 350 kHz can be achieved using the simple timing model.

The maximum column period can be computed by summing the total number of serializer clock cycles and then multiplying by the serializer clock period. The number of clock cycles required include: the downlink data packet components (zip code, pixel data words, and wait states) plus the "last data word to column strobe", the column pulse width and any time required by the controller board. In simple timing mode, the "last data word to column strobe" is the propagation delay of the downlink data packet (relative to the column strobe propagation). In advanced timing mode, the "last data word to column strobe" represents the propagation delay of just the input registers in the SLM212 Driver. There is likely some time required by the controller board. This is dependent on the controller board design, however, an estimate is provided in the tables below. The possible reasons for controller board time consumption is listed in note 2 below.

In the simple timing model, time must also be allocated to allow the data packet to propagate through the downlink components while waiting to pulse the column strobe. The computation of the maximum column rate for both timing models is show in Table 3-10.

	Simple Timing	**Advanced Timing**	
Amplitude Mode	Serializer Clock Cycles	Serializer Clock Cycles	Units/ (Notes)
Zip Code	2	2	
Pixel Data Cycles	**272**	**272**	
Wait States	4	4	
Last Data Word to Column Strobe	26	8	(1)
Column Pulse Width	1	1	
Controller Allocation	3	3	(2)
Total Clock Cycles	**308**	**290**	
Serializer Clock Frequency	**108**	**108**	MHZ
DCLK Period	9.26	9.26	nsec
COL Period	2.85	2.69	usec
Column Rate	**350.6**	**372.4**	kHz

Table 3-10 Maximum Column Rate (all 1088 pixels)

The actual timing specifications that must be met are the maximum serializer clock and the column to packet timing requirements listed in Table 3-12 and Table 3-13. Therefore, if the user only operates a subset of the pixels, then higher column rates can be achieved. Table 3-11 shows the maximum column rate possible when only operating 544 pixels. In this case the each driver would support 136 pixels. Further information on operating with a subset of pixel can be obtained by contacting Silicon Light Machines.

	Simple Timing	Advanced Timing	
Amplitude Mode	Serializer Clock Cycles	Serializer Clock Cycles	Units/ (Notes)
Zip Code	2	2	
Pixel Data Cycles	**136**	**136**	
Wait States	4	4	
Last Data Word to Column Strobe	26	8	(1)
Column Pulse Width	1	1	
Controller Allocation	3	3	(2)
Total Clock Cycles	**172**	**154**	
Serializer Clock Frequency	**108**	**108**	MHZ
DCLK Period	9.26	9.26	nsec
COL Period	1.59	1.43	usec
Column Rate	**627.9**	**701.3**	kHz

Table 3-11 Maximum Column Rate for 544 Pixels

The timing specifications for the timing interface on the module are shown in the in the Timing Specification in the following section. Timing specifications are also provided for the SLM212 driver chip in Section 5.6.

Notes for Tables:

(1) Last Data Word to Col for simple timing mode is 24 clock + 18 ns. At 108 MHz, 26 clocks are allocated.

(2) Time allocated for controller board: The number of clock cycles should be determined by the user based on the controller board design. Time may need to be allocated for: (a) controller state machine to detect column strobe and start outputting the data packet; (b) some propagation delays in the controller board's FPGA of the download packet data (c) time to synchronize the incoming col strobe if column strobe was generated in a different clock domain, (d) etc.

Timing Specification

Column to Packet Timing - Simple Timing Model

In the simple timing model, the next downlink packet is output after the COL strobe is pulsed. The COL can be pulsed after the last word of the downlink packet plus the time that it takes the downlink packet to propagate through the downlink path (i.e. serializer, deserializer, FPGA and input registers of the SLM212 driver). The downlink packet is shown at the input to the serializer. The column strobe is shown at the location of the cable.

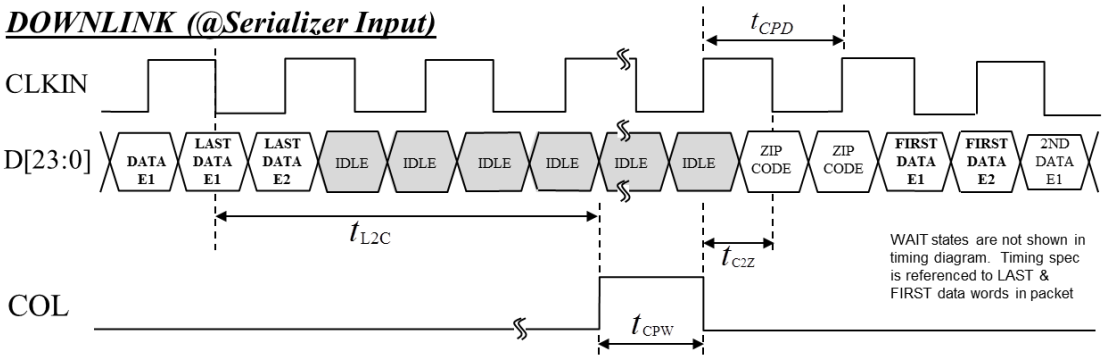

Figure 3-15 Packet to Column Timing Diagram – Simple Model

Parameter	Description	Min	Max
t_{L2C}	Last Data Word to Column	24 x t_{CPD} + 18 nsec	-
t_{C2Z}	Column to Zip Code	0 nsec	-
t_{CPW}	Column Pulse Width	7 nsec	-
F_{ser}	Serializer Clock Input Frequency	66 MHz	108 MHz

Table 3-12 Packet to Column Timing Specification – Simple Model

- After last data word in downlink packet; wait for t_{L2C} prior to outputting COL strobe
  - Zip code for next packet can immediately follow COL strobe

- Downlink data is shown at the input to the serializer on the controller board.
  - COL is shown on the cable.

Column to Packet Timing - Advance Timing Model

In the advanced timing model, the next downlink packet can be output before the COL strobe is pulsed for the previous packet. The COL can be pulsed after the last word of the downlink packet plus the time that it takes the downlink packet to propagate through the downlink path (i.e. serializer, deserializer, FPGA and input registers of the SLM212 driver). The downlink packet is shown at the input to the serializer. The column strobe is shown at the location of the cable.

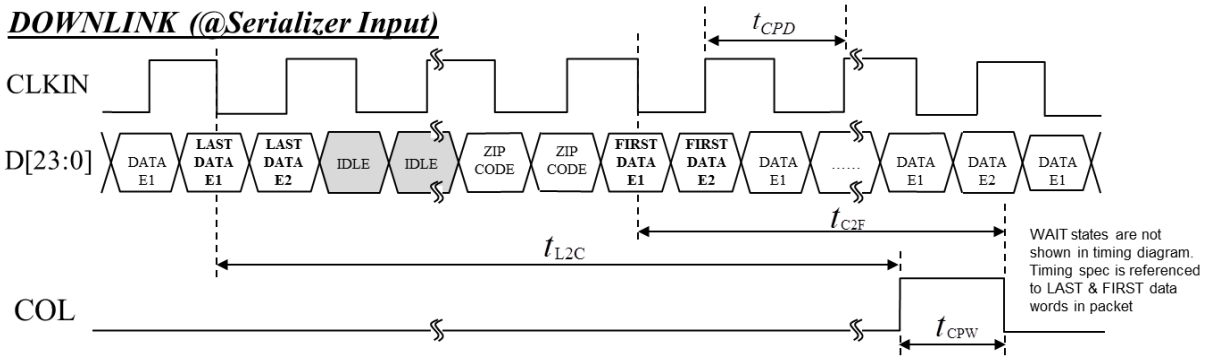

Figure 3-16 Packet to Column Timing Diagram – Advanced Model

Parameter	Description	Min	Max
t_{L2C}	Last Data Word to Column	(24 x t_{CPD}) + 18 ns	-
t_{C2F}	Column to First Data Word	-	19 x t_{CPD} – 3 ns
t_{CPW}	Column Pulse Width	7 nsec	-
F_{CPD}	Serializer Clock Input Frequency	66 MHz	108 MHz

Table 3-13 Packet to Column Timing Specification – Advanced Model

- Advanced timing may increase column rates by up to 6% (estimate)
- After last data word in downlink packet wait for tL2C prior to outputting COL strobe.
- Next packet can be output prior to COL strobe due to propagation delay of downlink packets through the serializer, deserializer, FPGA & SLM212 driver.

- Downlink data is shown at the input to the serializer on the controller board.
- COL is shown on the cable.

3.2.3 Control Interface (I^2C)

The control interface is implemented with an I^2C bus. The I^2C bus consists of 2 bi-directional signals, including a clock (SCL) and a data signal (SDA). The I^2C bus is implemented on the Cricket Board with an I^2C bus buffer chip (NXP p/n: P82B96). The buffer allows for operation with longer cables. It is recommend that the user also implement a P82B96 I^2C bus buffer chip on the controller board with 300 ohm pull up resistors to 5 volts for both SDA and SCL. Figure 3-17 illustrates the implementation of the I^2C bus over a cable with a P82B96 I^2C bus buffer on each side of the cable.

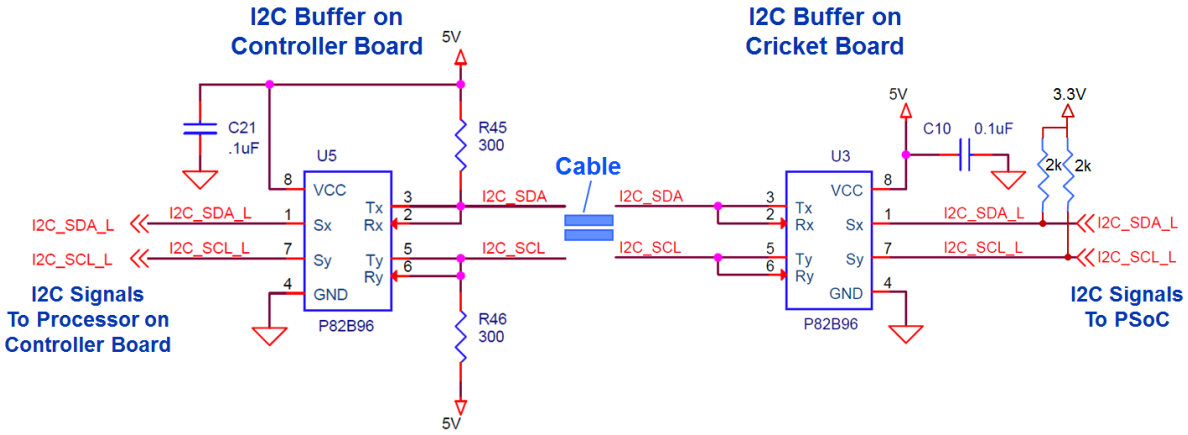

Figure 3-17 Using I^2C Buffer to Implement I^2C Bus across Cable

3.2.3.1 I2C Slave Address and Clock Rate

The PSoC on the High-Speed Buddy module is designed to be an I^2C slave. Its slave address is 0x29 (41). The I^2C master sends command and data bytes to the PSoC to control the PSoC operations on the High-Speed Buddy module. The user's controller board should implement an I^2C master. The I^2C master should support the I^2C clock stretching feature defined in the I^2C specification.

The PSoC I^2C interface supports up to 80 KHz clock rate. The I^2C bus master shall be configured to run with a bus clock of 80 KHz or less.

3.2.3.2 I2C Packet Format

The I^2C specification defines how data is transferred on the I^2C bus. The data packet starts with a control byte (i.e. start byte) which is composed of a 7-bits slave address and the least-significant bit is a Read/Write Bit. Any data that follows is user defined. For the High-Speed Buddy, the control byte is followed by a command byte. The command byte defines the function of the data packet. Data that follows the command byte is function dependent. The packet format used for the High-Speed Buddy is illustrated in Figure 3-18.

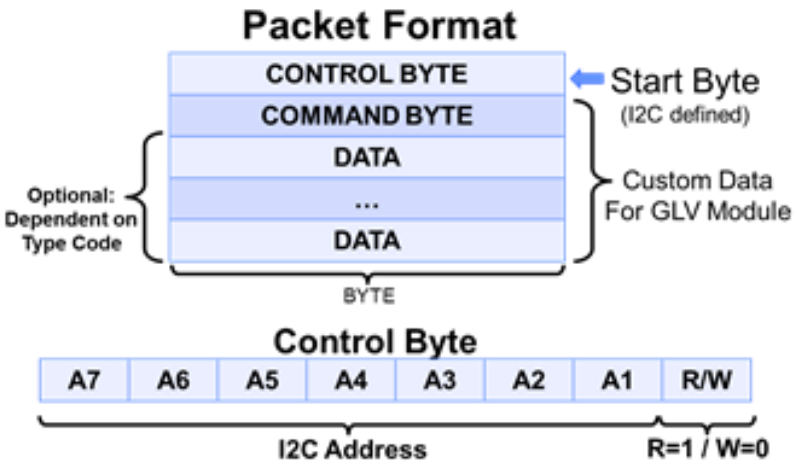

Figure 3-18 I²C Packet Format

3.2.3.3 Command Byte and User Functions

The first byte sent from the I²C master to the PSoC slave, right after the slave address byte (control byte), is processed by the PSoC slave as the command byte. This command byte tells the PSoC slave to perform one of the functions programmed into the PSoC chip (Module Device Control, A/D conversion, D/A conversion, EEPROM read and write). Figure 3-19 shows the command byte format (bit assignment) and defines the three function bits [F2:F0].

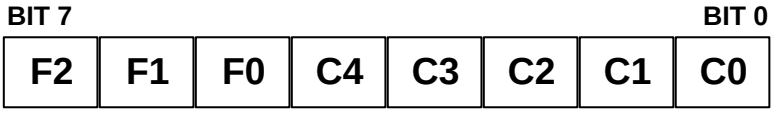

Figure 3-19 PSoC I²C Command Byte Format

Bit F2	Bit F1	Bit F0	Type of Command Function	Bits C4-C0
0	0	0	EEPROM read and write operation	Sector #
0	0	1	Reserved	
0	1	0	D/A conversion operation	Channel #
0	1	1	Reserved	
1	0	0	A/D conversion operation	Channel #
1	0	1	Reserved	
1	1	0	Get PSoC status	Data Type
1	1	1	Module Device Control	Control Option

Table 3-14 PSoC I²C Commands

And bits C4, C3, C2, C1 and C0 define the A/D or D/A channel number, or EEPROM sector (bank) number, or Module Device Control option, depending on the type of command function specified in bits F2, F1 and F0 (see Table 3-14).

3.2.3.4 EEPROM Write Protocol

The EEPROM write protocol allows the user to write data to the EEPROM located on the PSoC. There are 4 kBytes of EEPROM data that is organized in 16 banks of 256 bytes (Banks 0 to 15). The user is allowed to write to Banks 0 to 7. When the High-Speed Buddy Module is shipped; Banks 0 to 7 are empty and the user is free to write any user data to these banks. Banks 8 to 15 contain configuration and calibration data and the user should not write to these banks.

The I^2C bus master sends the following bytes (Table 3-15) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte> <offset-byte> <data-byte-1> <data-byte-2> … <data-byte-N>

The bus master shall wait 60 milliseconds (for actual EEPROM write operation to complete) before starting to read back the status byte.

The I^2C bus master starts a data read operation. It reads one (1) data byte from the PSoC via I^2C bus. This data byte indicates the status of the EEPROM write operation (success or failure, see Table 3-16). For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0x00 | EE-sector#)	EEPROM has 16 sectors. Each sector has 256 bytes. The user write-able sector number is from 0 to 7.
Offset Byte	Must be 0, 64, 128 and 192	This byte places the internal address counter at this offset within the selected sector. New data bytes are written to the EEPROM starting at this location.
Data Bytes	0 to 255	Number of data bytes must be exactly 64.

Table 3-15 EEPROM Write Command Format

Status Code	Description
1	Write status is read before the write completion (wait time is too short).
0	Successful completion of the EEPROM-write operation.
-1	Unsuccessful completion (most likely due to flash protection bit errors).
-2	Stack space was not sufficient for algorithm requirements (for debugging only)
-3	Write address is outside of the EEPROM range.
-4	Write address is not at the block (64-byte) boundaries.
-5	Data length (in bytes) to be written is not 64.

Table 3-16 EEPROM Write Status Codes

3.2.3.5 EEPROM Read Protocol

The EEPROM read protocol allows the user to read data from the EEPROM on the PSoC. Banks 0 to 7 are empty when the module is delivered. Banks 8 to 15 are configured in accordance with the EEPROM memory map shown in section 8 "Appendix E - EEPROM Data Specifications".

The I^2C bus master sends the following bytes (Table 3-17) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte> <offset-byte>

The bus master shall wait one millisecond before starting the actual data read operation.

The I^2C bus master starts a data read operation. It reads data bytes from EEPROM via I^2C bus one byte at a time until all *N* bytes are received (*N > 0, and N + OffsetByte < 257*). For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0x00 | EE-sector#)	EEPROM has 16 sectors. Each sector has 256 bytes. The read-able sector number is from 0 to 15.
Offset Byte	0 to 255	This byte places the internal address counter at this offset within the selected sector. The bus master will be reading from EEPROM starting at this location.

Table 3-17 EEPROM Read Command Format

User may retrieve data from and store data to the PSoC EEPROM via I^2C at any time after supplying 5.0V power to the PSoC (before and after powering up the module). See Section 8 for Appendix E - EEPROM Data Specifications.

3.2.3.6 D/A Conversion Protocol

The D/A conversion protocol allows the user to set or adjust the following programmable voltages to the GLV and drivers: high-voltage (Vddah), bias and common. The user uses a 9-bit digital DAC value to set the voltage. The user can compute the associated voltage generated from the digital value using the equation described in this section.

The I^2C bus master sends the following bytes (Table 3-18) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte> <data-word-high-byte> <data-word-low-byte>

The bus master shall wait two milliseconds (for D/A operation to complete) before starting to read the status byte.

The I^2C bus master starts a data read operation. It reads one (1) data byte from the PSoC via I^2C bus. This data byte indicates the status of the D/A conversion (success or failure, see Table 3-19). For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0x40 | channel#)	The 9-bit D/A converter has 3 channels; the channel number is from 0 to 2 (see Table 3-20).
Data Word High Byte	0 to 1	High byte of 16-bit data word which only uses its lower 9-bits (it is right-justified, and ranging from 0 to 510).
Data Word Low Byte	0 to 255	Low byte of 16-bit data word which only uses its lower 9-bits (it is right-justified, and ranging from 0 to 510).

Table 3-18 D/A Conversion Command Format

Status Code	Description
1	D/A conversion is in-progress (not completed yet).
0	Successful completion of D/A conversion.
-1	Digital data sent to PSoC D/A is over the upper limit (EEPROM limiter, <=510)
-2	Digital data sent to PSoC D/A is below the lower limit (0).
-3	D/A channel number is invalid.

Table 3-19 D/A Conversion Status Codes

D/A Channel#	Analog Output	$Analog_{min}$ to $Analog_{max}$	$Data_{min}$ to $Data_{max}$	Conversion (Equation 3-1)
0	VDDAH (volts)	13 to 24 volts	0 to 510	LINEAR
1	Common (volts)	0 to 24 volts	0 to 510	LINEAR
2	Bias (volts)	0 to 24 volts	0 to 510	LINEAR

Table 3-20 D/A Conversion Channel Assignment

Note that the maximum voltages for Common and Bias are limited by the VDDAH voltage.

User may set the desired VDDAH (or Common or Bias) voltage to the GLV by sending the D/A-Conversion command byte (0x40 | D/A-channel-number) to PSoC via I^2C along with 2 bytes of D/A channel input data. Then read the D/A conversion status byte back from the I^2C slave. The D/A input data for the VDDAH channel is computed using the following equation (Equation 3-1) where $VDDAH_{max}$ is the analog output voltage when the D/A input is at the maximum (510), and $VDDAH_{min}$ is the analog output voltage when the D/A input is 0 (also see Table 3-20). The D/A input data for the Common and Bias channels can be derived in the same way. And these D/A channel calibration data (i.e. Analog-min, Analog-max, Data-min & Data-max) can be retrieved from the PSoC EEPROM (see Section 8.5).

$$DAC_{\text{data}} = \text{INT}\left[510 \cdot \frac{(VDDAH - VDDAH_{\min})}{(VDDAH_{\max} - VDDAH_{\min})} \right]$$

Equation 3-1 PSoC D/A Conversion for VDDAH

3.2.3.7 A/D Conversion Protocol

The A/D conversion protocol allows the user to monitor critical power supply voltages and currents on the High-Speed Buddy. In addition, the temperature on the High-Speed Buddy module can be read. The PSoC will return an 11-bit digital value in response to the A/D conversion protocol. The user can compute the associated voltages, currents and temperature from the digital status word by using the equation described in this section.

The I^2C bus master sends the following bytes (see Table 3-21) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte>

The bus master shall wait 50 milliseconds (for the data sampling to complete) before starting to read the actual A/D conversion data.

The I^2C bus master starts a data read operation. It reads two (2) data bytes from the PSoC via I^2C bus. The two bytes hold the digital data acquired from the A/D converter, which is 11-bit signed (from –1024 to +1023) and right-justified (using the lower 11-bits). For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0x80 | channel#)	The 11-bit A/D converter has 17 channels; the channel number is from 0 to 15 (see Table 3-22).

Table 3-21 A/D Conversion Command Format

A/D Channel#	Analog Input	$Analog_{min}$ to $Analog_{max}$	$Data_{min}$ to $Data_{max}$	Conversion (Equation 3-2)
0	5V (volts)	0 to 7.8	-1024 to 1023	LINEAR
1	5V Current (mA)	0 to 4814.8	-1024 to 1023	LINEAR
2	VH (volts)	0 to 28.6	-1024 to 1023	LINEAR
3	VH Current (mA)	0 to 1181.8	-1024 to 1023	LINEAR
4	VDDx (volts)	0 to 5.2	-1024 to 1023	LINEAR
5	VDDx Current (mA)	0 to 1181.8	-1024 to 1023	LINEAR
6	VDDAH (volts)	0 to 28.6	-1024 to 1023	LINEAR
7	VDDA3.3 (volts)	0 to 5.2	-1024 to 1023	LINEAR
8	Common (volts)	0 to 28.6	-1024 to 1023	LINEAR
9	Bias (volts)	0 to 28.6	-1024 to 1023	LINEAR
10	TOUT (volts)	0 to 2.6	-1024 to 1023	LINEAR
11	TC0 U1 (volts)	0 to 28.6	-1024 to 1023	LINEAR
12	TC0 U2 (volts)	0 to 28.6	-1024 to 1023	LINEAR
13	TC0 U3 (volts)	0 to 28.6	-1024 to 1023	LINEAR
14	TC0 U4 (volts)	0 to 28.6	-1024 to 1023	LINEAR
15	VDDA Current (mA)	0 to 1181.8	-1024 to 1023	LINEAR
16	Temperature (Celsius)	-40 to 150 C	-160 to 600	LINEAR

Table 3-22 A/D Conversion Channel Assignment

User may read the 11-bit A/D conversion data from any of the 17 PSoC A/D channels by sending the A/D-Conversion command byte (0x80 | A/D-channel-number) to the PSoC via I^2C. The 11-bit A/D data can be converted to the corresponding analog value using the following equation (Equation 3-2) where Data-max is the 11-bit A/D data value that represents the Analog-max value, and Data-min is the A/D data value that represents the Analog-min value (also see Table 3-22). This A/D channel calibration data (i.e. Analog-min, Analog-max, Data-min & Data-max) can be retrieved from the PSoC EEPROM Bank 9 (see Section 8.4).

$$Ana\,log_{\,\text{value}} = Ana\,log_{\,\text{min}} + \left(\frac{ADC_{\text{data}} - Data_{\text{min}}}{Data_{\text{max}} - Data_{\text{min}}} \right) \cdot (Ana\,log_{\,\text{max}} - Ana\,log_{\,\text{min}})$$

Equation 3-2 PSoC A/D Conversion

3.2.3.8 Read PSoC Status Protocol

The PSoC status protocol allows the user to read status information from the PSoC, including processor status, power supply status, error codes and the PSoC firmware version number. The PSoC will return a one byte status in response to the PSoC status protocol. The user can decode the status information using the tables listed in this section.

The I^2C bus master sends the following bytes (see Table 3-23) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte>

The bus master shall wait one millisecond (for the data acquisition to complete) before starting to read the status data.

The I^2C bus master starts a data read operation. It reads one (1) data byte (status data, see Table 3-24) from the PSoC via I^2C bus. For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0xC0 | DataType)	For data type definitions, see Table 3-24.

Table 3-23 Command Format for Reading PSoC Status

Data Type	Type of Status Data Returned
0	PSoC Processor Status and Control Register value (see Table 3-25)
1	GLV Driver Power-on Status Register value (see Table 3-26)
2	Reserved
3 to 25	TBD (always return a value of zero)
26	Source ID of the last reported PSoC error since reset (see Table 3-27)
27	PSoC error code last reported from an error source (see Table 3-27)
28	PSoC Firmware Major Revision Number (0 to 255)
29	PSoC Firmware Minor Revision Number (0 to 255)
30	High Byte of PSoC Firmware's Revision-Control Check-in ID (0 to 255)
31	Low Byte of PSoC Firmware's Revision-Control Check-in ID (0 to 255)

Table 3-24 Data Types used in the Command Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Global Interrupts: 1=Enabled 0=Disabled	Not used	Watchdog Reset: 1=Occurred 0=No WDR	Power-On Reset: 1=Occurred 0=No POR	CPU Sleep: 1=Sleep 0=Normal	Not used	Not used	CPU Stop: 1=Halt 0=Normal

Table 3-25 PSoC Processor Status and Control Register

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GLV Driver 1=shutdown 0=normal	LVDS RX_48 1=ON 0=OFF	LVDS RX_2 1=ON 0=OFF	Not Used	Not used	VDDAH 1=ON 0=OFF	VDDA3.3 1=ON 0=OFF	VDDX3.3 1=ON 0=OFF

Table 3-26 GLV Driver Power-on Status Register

Key:

- LVDS RX_48: DS90CR286 Deserializer for Downlink
- LVDS RX_2: FIN1048 LVDS Receiver for COL & WCLK

Source ID	Error Source	Error Codes
0	None	None
11	PSoC Generic	None
12	A/D Conversion	None
13	D/A Conversion (channel 0)	See Table 3-19 D/A Conversion Status Codes
14	D/A Conversion (channel 1)	See Table 3-19 D/A Conversion Status Codes
15	D/A Conversion (channel 2)	See Table 3-19 D/A Conversion Status Codes
16	Reserved	Reserved
17	EEPROM write	See Table 3-16 EEPROM Write Status Codes
18	I^2C Interface	0 = OK; -1 = undefined command; -2 = empty command; -3 = data transfer error.
19	Module Lifetime Store	0 = OK; -1 = error storing data; -2 = stack overflow (for debugging); -3 = bad checksum after lifetime store
20	Power Up/Down	See Table 3-30 Module Device Control - Status Codes
21	Data Reg. Read	None

Table 3-27 PSoC Error Sources and Their Error Codes

3.2.3.9 Module Device Control Protocol

The Module Device Control protocol allows the user to perform critical initialization functions on the High-Speed Buddy, including: powering up and powering down the module set, de-skewing the deserializer and resetting the Cricket FPGA.

The I^2C bus master sends the following bytes (see Table 3-28) to the PSoC, in accordance with Section 7.1 [Sending Data from I2C Bus Master to Slave]:

<control-byte> <command-byte>

The bus master shall wait 500 milliseconds (for the Module Device Control function to complete) before starting to read the status byte. User shall poll the status byte at the end of this 500-millisecond wait period.

The I^2C bus master starts a data read operation. It reads one (1) data byte (Module Device Control status, see Table 3-30) from the PSoC via I^2C bus. For details see Section 7.2 [Receiving Data from Slave by I2C Bus Master].

See Section 3.3 for details on how to properly power up and initialize the High-Speed Buddy.

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the PSoC slave address; its LSB is 0 for write operation.
Command Byte	(0xE0 | PowerCntrl)	PowerCntrl defines the control options. Control options are listed in Table 3-29

Table 3-28 Command Format for Module Device Control

Control Option	Description
0	Power down all voltages
1	Power up all voltages
2	Power up low voltages only
3	Turn off LVDS de-skew function
4	Turn on LVDS de-skew function
5	Reset the FPGA

Table 3-29 Module Device Control – Control Options

Status Code	Description
1	GLV Power up/down operation is in-progress (not completed yet).
0	Successful completion of GLV Power up/down operation.
-1	VDDAH is not up after power-up, or is not down after power-down.
-2	VDDA3.3 is not up after power-up, or is not down after power-down.
-3	User has shut down VDDAH via HV_EN pin on J1 connector
-4	Low Voltage Detection interrupt has been triggered on PSoC VDD (PSoC supply fell below the operating threshold)
-5	VH (24V) power booster output fell below the operating threshold
-6	V5 (5V) power supply fell below the operating threshold
-7	Invalid GLV driver Power up/down option
-8	FPGA initialization has not completed
-9	FPGA CRC error occurred during initialization

Table 3-30 Module Device Control - Status Codes

3.2.4 Other Signals

The other signals on the J1 I/O connector are listed below:

- RES Input
- HV_EN Input
- I2C_RDY Output
- TOUT_A Output (analog)
- SPARE[3:0] TBD

RES: Is the input signal that resets the PSoC.

HV_EN: Control signal that allows the user to enable or disable the high-voltage to the GLV and drivers. When HV_EN is LOW then the high-voltage to the GLV and drivers will be turned off. When HV_EN is HIGH, then the high-voltage will be applied to the GLV and drivers after the I2C "PowerUp" command has been executed.

I2C_RDY: Status signal that indicates the PSoC is ready to receive commands on the I^2C bus. After 5V power has been applied to the Cricket Board the I2C_RDY signal will initially be de-asserted (LOW). After the PSoC has booted-up, has initialized the I^2C slave circuitry and is ready to receive I^2C commands, then the PSoC will assert the I2C_RDY signal "HIGH". The controller should wait until I2C_RDY is HIGH until any I^2C commands are written to the PSoC.

TOUT_A: Analog test output signal. The user can monitor any one of the 16 analog channels (channels 0 to 15) of the PSoC A/D function. Only the temperature channel is not available since the temperature sensor is a digital sensor. To select a channel, send an I^2C "A/D conversion protocol" packet.

SPARE[3:0]: Four spare signals are reserved for future growth. These signals are connected to the FPGA on the Cricket Board. These signals may have test signal outputs assigned from the Cricket FPGA. On the controller board, these signals should also be connected to the FPGA for future growth and assigned as unused inputs or tri-stated pins.

3.2.5 Power Input

The module operates with a single 5 volt power input. The combined power consumption for the High-Speed Buddy Module and Cricket Interface Board is 10 watts (typical) and 15 watts (max). In response, to a I^2C "PowerUp Low" or "PowerUp" command, surge currents may occur when the Cricket's voltage regulators are turned on.

3.3 Module Initialization

This section discusses the procedures to power up and initialize the High-Speed Buddy. In addition, the power down procedure is described.

3.3.1 Power Up and Initialization Sequence

Table 3-31 shows the power up and initialization sequence for the High-Speed Buddy Module.

Step	Other	I2C			Downlink Interface	Column Strobe	Wait After	Comments
		Read/ Write	Function (F2-F0)	Code (C4-C0)				
1	Turn on 5 Volt power supply						2 sec (3)	Power to Cricket Board's power connector
2	Pulse Reset (RES) Signal High						-	RES signal is on Cricket Board's I/O connector and drive the XRES pin of the PSoC. Pulse width of active high RES signal should be 10 usec (min).
3	Wait for "I2C_RDY = 1"						-	I2C_RDY signal is on Cricket Board's I/O connector
4a		Write	F2:F0 = 7 Device Control	C4:C0 = 2 PowerUp Low			500 msec	Serializer clock input should be stable. Do not change clock frequency after powerup low.
4b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
5	Serializer Pin DS_OPT = LOW							Serializer/Deserializer used in configuration 5 where serializer inputs BAL=High & CON1=High (2)
6a		Write	F2:F0 = 7 Device Control	C4:C0 = 4 Deskew ON			500 msec (3)	Deskew the Deserializer
6b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
7a		Write	F2:F0 = 7 Device Control	C4:C0 = 3 Deskew OFF			500 msec (3)	Serializer is operated during deskew.
7b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
8	Serializer Pin DS_OPT = HIGH						-	DS_OPT is input pin of DS90CR485 serializer
9a		Write	F2:F0 = 7 Device Control	C4:C0 = 5 Reset FPGA			500 msec (3)	
9b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
10					Initalize SLM212 registers		-	Run 11 step initialization sequence for SLM212 Drivers
11					Write all pixels with amplitude = 0		(1)	Write to Amplitude 0 register array
12						Pulse	(1)	Note: See packet to COL timing specification
13a	Set "HV_EN = 1"						-	HV_EN is an input to the HS-Buddy on the I/O cable
13b		Write	F2:F0 = 7 Device Control	C4:C0 = 1 PowerUp			500 msec	
13c		Read	Status Byte				-	If Status = 0 THEN GOTO next step
14a		Write	F2:F0 = 2 D/A Conversion	C4:C0 = 2 Bias			2 msec	Typical value = 0
14b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
15a		Write	F2:F0 = 2 D/A Conversion	C4:C0 = 1 Common			2 msec	Typical value = 0
15b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
16a		Write	F2:F0 = 2 D/A Conversion	C4:C0 = 0 Vddah			2 msec	Each module will have a recommended value
16b		Read	Status Byte				-	If Status = 0 THEN GOTO next step
LOOP					Write pixel amplitude values		(1)	Operate by sending pixel amplitude values to SLM212 Driver's Amplitude #0 register array
						Pulse		

Table 3-31 Power Up and Initialization Sequence

Notes: (1) See Downlink Packet to Column Strobe Timing Specification
(2) See datasheet for Serializer DS90CR485 & Deserializer DS90CR486
(3) Wait time will be optimized during performance testing.

Step 1: Turn on the 5 volt power supply.

Step 2: Reset the PSoC by asserting the RES signal on I/O connector. The pulse width of active high RES signal should be 10 usec (min). The RES signal will drive the XRES signal input pin of the PSoC and will reset the PSoC. The PSoC will boot-up. The PSoC will enable the I^2C bus slave function and then assert the I2C_RDY status signal on the I/O connector.

Step 3: The controller board should wait until I2C_RDY is asserted by the PSoC.

Step 4a: On the I^2C Bus, the I^2C master should send a "PowerUp Low" command (command byte=0xE2). In response, the PSoC will turn on all the supply low voltages on the Cricket Board and High-Speed Buddy Module. Before sending the Power Up Low command, the serializer clock input should be locked and stable. Do not change the frequency of the clock after the PowerUp Low command has been sent.

Step 4b: After sending the I^2C command, read the "Module Control Device" Status byte via I^2C. If Status = 0 then go to the next step.

Step 5: To start the deskew process between the serializer and the deserializer, the controller should reset the DS_OPT pin of the DS90CR485 serializer to "LOW".

Step 6a: On the I^2C Bus, the I^2C master should send a "Deskew ON" command (command byte=0xE4). In response, the PSoC will reset the deserializer' DESKEW pin to "LOW". The deskew process is now active. The serializer on the controller board must be operated during the deskew operation. For proper deskew procedure, please refer to National Semiconductor Data Sheets: DS90C486 and DS90C485.

Step 6b: After sending the I^2C command, read the "Module Control Device" Status byte via I^2C. If Status = 0 then go to the next step.

Step 7a: On the I^2C Bus, the I^2C master should send a "Deskew OFF" command (command byte=0xE3). In response, the PSoC will set the deserializer' DESKEW pin to "HIGH".

Step 7b: After sending the I^2C command, read the "Module Control Device" Status byte via I^2C. If Status = 0 then go to the next step.

Step 8: To conclude the deskew process between the serializer and the deserializer, the controller should set the DS_OPT pin of the DS90CR485 serializer to "HIGH".

Step 9a: On the I^2C Bus, the I^2C master should send a "Reset FPGA" command (command byte=0xE5). In response, the PSoC will reset the Cricket FPGA.

Step 9b: After sending the I^2C command, read the "Module Control Device" Status byte via I^2C. If Status = 0 then go to the next step.

Step 10: Using the downlink interface, initialize the SLM212 drivers. The 11 step initialization sequence for the SLM212 drivers is illustrated in Table 3-5 and Table 3-6.

Step 11: Using the downlink interface, initialize all pixels values to 0. This is done by writing to the SLM212 driver's Amplitude 0 register arrays. This is procedure is illustrated in Table 3-4.

Step 12: Pulse the Column Strobe signal high for 10 ns (min). This will update all SLM212 driver channel High-Voltage output to the zero amplitude value loaded in step 11.

Step 13a: Set the HV_EN signal to "HI". This will allow the forthcoming High-Voltages to be applied to the GLV and drivers.

Step 13b: On the I^2C Bus, the I^2C master should send a "PowerUp" command (command byte=0xE1). In response, the PSoC will power up the high voltage power supply to the minimum active setting of 13 volts. Note that all three D/A converters on the PSoC are reset to the minimum settings during the GLV power-up.

Step 13c: After sending the I^2C command, read the "Module Control Device" Status byte via I^2C. If Status = 0 then go to the next step.

Step 14a: On the I^2C Bus, the I^2C master should send a Bias "D/A conversion" command (command byte=0x42). In response, the PSoC will set the Bias DAC to the user programmed level. A typical value for Bias is zero volts.

Step 14b: After sending the I^2C command, read the "D/A conversion" Status byte via I^2C. If Status = 0 then go to the next step.

Step 15a: On the I^2C Bus, the I^2C master should send a Common "D/A conversion" command (command byte=0x41). In response, the PSoC will set the Common DAC to the user programmed level. A typical value for Common is zero volts.

Step 15b: After sending the I^2C command, read the "D/A conversion" Status byte via I^2C. If Status = 0 then go to the next step.

Step 16a: On the I^2C Bus, the I^2C master should send a High-Voltage "Vddah" "D/A conversion" command (command byte=0x40). In response, the PSoC will set the Vddah DAC to the user programmed level. The recommended maximum setting for Vddah DAC is stored in the PSoC EEPROM Bank 10, Entry 1, address 0x02-0x03 (Vddah error level). A Vddah error level is a voltage limiter value. The PSoC on the High-Speed Buddy module will prevent users from requesting a Vddah setting on the I^2C interface that is higher than this Vddah limiter value.

Step 16b: After sending the I^2C command, read the "D/A conversion" Status byte via I^2C. If Status = 0 then go to the next step.

After completing the 16 step initialization process, the module is ready for operation. The user can start sending pixel amplitude values by writing to the driver's Amplitude 0 register arrays over the downlink interface. The driver Amplitude 0 values are updated to the GLV ribbons when the COL strobe is asserted.

3.3.2 Power Down Sequence

It is recommended that before the 5 volt power supply is turned off, the controller shall power down the module by sending the GLV Power-Down command byte (0xE0) to the PSoC via I^2C, and read the power-down status byte via I^2C.

However, as an alternative, the user may simply turn off the power supplies to the module when they are ready to shut down the module. The PSoC constantly monitors the power supplies to the module, when the PSoC detects a low voltage condition then the PSoC disables the high voltage regulator (VDDAH) to the GLV and SLM212 drivers instantly to prevent damage to the GLV.

VDDAH can be shut down from 5 different sources. Two dedicated PSOC hardware comparators can independently and automatically shut down VDDAH when a low voltage condition is detected the input voltage or if the VDDAH voltage generation circuit fails. The PSOC software operates a third shutdown control. A fourth shutdown control is the HV_EN signal that is an input on the J1 connector (pin B20) and allows the user to shut down the high voltage with a logic low. This HV_EN signal is also connected to a pull-up resistor to allow correct operation when pin J1-B20 is left unconnected. Any of the four shutdown controls can turn off VDDAH. The VDDAH must be shutdown prior to reaching the voltage at which devices on the module stop functioning. Since the comparators can shutdown VDDAH without software intervention, the shutdown process can happen within 20 microseconds of the input voltage crossing the shutdown threshold.

4 Appendix A – SLM212 Driver: Modes of Operation

The SLM212 supports the following modes:

- Operational Modes
- Power Modes
- Data Bus Modes

4.1 Operational Modes

The SLM212 supports the three modes of operation listed below. The user selects the mode by writing to the D1 bit "PWM-En" and the D9 bit "NoDelay" in the driver's Control register.

1. AMP Mode: (D1=0; D9=1)
2. AMP with Delay Mode: (D1=0; D9=0)
3. PWM Mode; (D1=1; D9=0)

Currently the High-Speed Buddy support only amplitude (AMP) mode of operation. It is possible for the High-Speed Buddy to support AMP with delay mode, if the user writes to the Delay 0 register array and transfers the data to the Delay 1 & 2 register arrays using the Delay Transfer register. Contact Silicon Light Machines for more information for the AMP with Delay mode. The PWM mode is fully supported with the standard Buddy Module.

Note: The standard Buddy Module was primarily designed to support the PWM mode of operation. The AMP Mode was a secondary mode of operation and was called the "Calibration Mode". Likewise, the AMP Mode with Delay was called the "Calibration with Delay" Mode.

4.1.1 AMP Mode

In the amplitude (AMP) mode, each of the 272 channels is loaded with a 10-bit value for each column of data. Once all 272 channels have been loaded, the COL signal is strobed to cause all the outputs to transition at the same time. Without delays all 272 outputs will transition to their new values immediately after the COL strobe. AMP mode requires much higher data rates at the input of the chip, but allows for continuous control of all 272 channels with the full 10-bit output range. For each COL cycle, 10-bits of data must be loaded into each of the 272 Amp0 registers before the COL signal is strobed.

4.1.2 AMP with Delay Mode

AMP with Delay Mode functions similar to the AMP mode, however, the time at which the output transitions to a new value can be delayed. Each of the 272 channels can be programmed independently so that each channel has unique delay. The delay values are set by writing to the eight bit Delay 0 Registers in each channel. The delay values are loaded into the Delay registers during driver initialization and are not intended to be changed during operation.

The new DAC value is output when the programmed delay for the given channel equals a global PWM counter value. The PWM counter is reset when the COL signal is asserted and increments on the rising edge of WCLK.

Delay 1 and Delay 2 Registers are not used in this mode. Since Amplitude 1 Register is used by the internal logic in the AMP with Delay Mode, then the user should not transfer data using the Data Transfer Register when operating in this mode. (Note: Both the GATE and P_EN control bits of the Data Transfer Register should both be initialized to one and should remain at this value when operating in the AMP with Delay Mode.)

It is recommended that a delay value of 128 be used as the nominal delay value. Delay values less than 128 should be used to reduce the delay and values greater 128 should be used to increase the delay from the nominal delay.

4.1.3 PWM Mode

In many applications the data is essentially monochrome and it is desired that each output switch between an on and off state. The PWM mode is designed with these applications in mind. Data is loaded as a single bit per pixel. Based on this binary data each channel outputs one of three pre-programmed levels. These three levels consist of an on state (AMP2), off state (AMP0), and an off-to-on transition state (AMP1). Each state is programmed, before the data stream begins, with a 10-bit output value. A representation of this mode is shown in Figure 4-1.

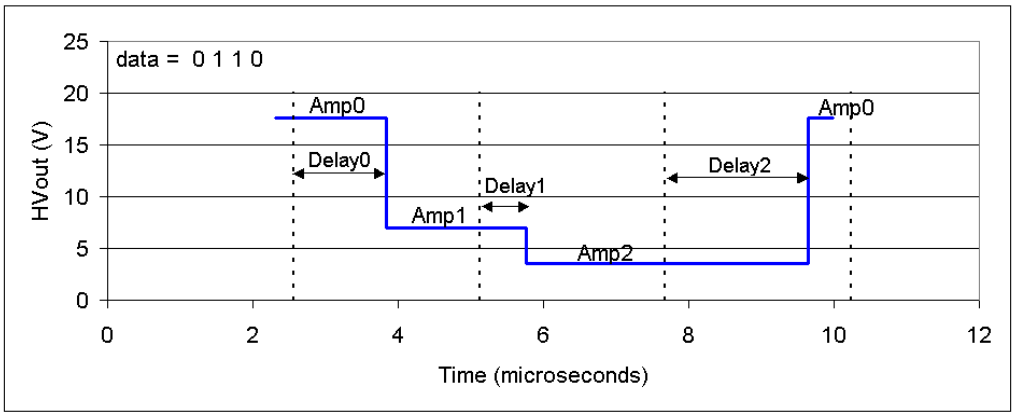

Figure 4-1 Example of PWM Mode

After each set of 272 bit data is loaded, the COL signal is strobed to output the new data. The actual output transition is delayed from the COL strobe by a programmed number of WCLK counts. Each of the three transitions has its corresponding delay, with the on state delay (DELAY2), off state delay (DELAY0) and the off-to-on transition state delay (DELAY1). The delay values are loaded into the DELAY registers during driver initialization and are not intended to be changed during operation.

It is recommended that a delay value of 128 be used as the nominal delay value. Delay values less than 128 should be used to reduce the delay and values greater 128 should be used to increase the delay from the nominal delay.

Figure 4-1 shows an example of the following register values: AMP0=1000, AMP1=400, AMP2=200, DELAY0=128, DELAY1=64, DELAY2=196. The figure also assumes the following values VDDAH=18 Volts, column period=2.56 usec and the PWM clock period=10 nsec.

4.2 Power Modes

Control register data bit D8 “PwrMode” allows for the selection of low power or constant power modes

4.2.1 Low Power Mode

The DACs current draw is minimized in order to minimize power dissipation. The only current generated is the current necessary for optimal high voltage output. (Using current steering technology, the complementary current is directed to the 3.3V VddM, instead of the 22V VddAH supply, thus low power dissipation achieved).

4.2.2 Constant Power Mode

For most operational conditions, the constant power mode is the preferred mode. In constant power mode, both the real output current and its complementary current are drawn from the 22V VddAH supply. Constant power dissipation and minimum cross-talk are achieved in constant power mode because the sum of these two currents is maintained at a constant value in every channel.

4.3 Data Bus Modes

The data bus can operate with two selectable options which are described below.

4.3.1 Bus Width Mode

The SLM212 driver can operate with a 10-bit data bus or a 20-bit data bus. The 20-bit data bus allows two pixel amplitude values (Amplitude #0 Register) to be written during a single bus cycle. Only the Amplitude 0 registers use the high order 10-bits (D[19:10] in the 20-bit data bus mode. The bus width is selectable by writing to the D4 bit “10-bit” in the driver’s Control register.

4.3.2 Count Up or Count Down Mode

Individual registers within register arrays are addressed by using the address counter. The address counter is initialized from the start address register. The address counter can count up or count down. The direction of the address counter is determined by the D0 bit “Up/Down” in the driver’s Control Register.

5 Appendix B – SLM212 Driver: Data Interface Description

The data interface for the SLM-212 driver chip is defined in this section. Table 5-1 summarizes the signals for the data interface. All signals listed are single ended, 3.3-volt logic levels.

5.1 Inputs and Outputs

The data interface consists of a 20-bit data bus, two control signals, and a data clock (DCLK). The control signals are an enable signal (EN) and Address Latch Enable (ALE). The data bus is a time-multiplexed data and address bus. A TOUTD pad can output either a parity signal or one of the parity circuit inputs.

In addition to the data interface, two signals are provided for timing control. The two signals are the column update signal (COL) and a high-speed PWM clock (WCLK). The pixel data is updated to drive the GLV pixels when the column strobe is asserted. The user is responsible for insuring that all pixel data transmission is completed between column strobes. The PWM clock determines the resolution of the pulse width in PWM mode or the pixel delay value for AMP with Delay mode. In normal AMP mode the PWM clock is not used and does not need to be supplied by the user. When operating in AMP mode, the PWM clock is not used.

Pin(s)	In/Out	Description
DCLK	In	Data clock used to clock EN, ALE, D[9:0].
EN	In	Enable when high indicates a valid zip code or data word is on the bus; when low, it indicates that the data bus is inactive.
ALE	In	Address latch enable when high indicates that a valid zip code is on the data bus and initiates a bus cycle.
D[19:0]	In	Data or zip code. This data bus supports a 10-bit mode.
TOUTD	Out	Digital test output bit which is multiplexed from serial parity, parallel parity, and digital test circuit.
WCLK	In	Pulse width modulation clock used to clock in the COL signal, drives the PWM circuitry, and drives the DAC output.
COL	In	Column strobe that resets the PWM counter, clocks in the pixel data for use in edge detection circuitry and provides a timing reference for all modes.

Table 5-1 Data Interface Signal Descriptions

The SLM212 driver can operate with a 10-bit data bus or a 20-bit data bus. High-Speed Buddy uses the 20-bit mode only. The bus size is selected by the D4 bit in the driver's Control register, therefore, D4 should be set to "0" to select the 20-bit data bus mode. The SLM212 driver supports a 20-bit data bus to allow writing two pixel amplitude values in one clock cycle. The full 20-bit bus is used only for writing to the driver's Amplitude #0 Register array. When writing pixel amplitude values using 20-bit bus mode, odd driver channels are written to the upper 10-bits D[19:10] while even channels are written to lower 10-bits D[9:0].

However, for all other SLM212 driver registers, only the lower 10 bits of the data bus is used, while the upper 10-bits are ignored. Likewise, the Zip Code is written to the lower 10-bits of the data bus. During the Zip Code cycle (ALE=1 & EN=1) the upper 10-bits are also ignored.

5.2 Registers and the Address Map

The data writes will support two types of internal registers: (1) single register; and (2) array of registers. Each register or array of registers is identified by a zip code (e.g. this is a group address or a single register address). Table 5-2 shows a list of the internal registers that can be written to via the data interface, the number of bits for each register, the number of registers and the assigned zip code. For registers that are actually an array of registers, Table 5-2 also lists the range of valid addresses that access the entire array. In addition, Table 5-2 defines if the register data is required for operating the High-Speed Buddy.

Register	Bits per Register	Type	Number of Registers	Zip Code D[9:0]	Register Address	Used on HS-Buddy
Amplitude 0	**10**	**Array**	**272**	**000h**	**[0:271]**	**Yes**
Pixel	8 x 1 bit	Array	34	003h	[0:33]	No
Delay 0	8	Array	272	004h	[0:271]	No
Slopes	**10**	**Array**	**16**	**007h**	**[0:15]**	**Yes**
Amplitude 1	10	Array	272	*	[0:271]	No
Amplitude 2	10	Array	272	*	[0:271]	No
Delay 1	8	Array	272	*	[0:271]	No
Delay 2	8	Array	272	*	[0:271]	No
Programmable ECREF	**10**	**Single**	**1**	**200h**	**0**	**Yes**
Current DAC 0	10	Single	1	201h	0	No
Current DAC 1	10	Single	1	202h	0	No
Analog Test	**10**	**Single**	**1**	**203h**	**0**	**Optional**
Start Address	**10**	**Single**	**1**	**204h**	**0**	**Yes**
Control	**10**	**Single**	**1**	**207h**	**0**	**Yes**
Data Transfer	10	Single	1	305h	GLOBAL	No
Digital Test	**10**	**Single**	**1**	**306h**	**0**	**Optional**

Table 5-2 List of Registers and Address Map

*The array registers Amplitude 1, Amplitude 2, Delay 1, & Delay 2 are not addressed directly using a zip code and starting address. These registers are written by copying from the Amplitude 0 or Delay 0 array registers according to bits in the Data Transfer register.

The High-Speed Buddy Module only supports AMP mode. Therefore, operation of the SLM212 drivers is greatly simplified. Only the following registers need to be written during operation: Amplitude 0 Register, Slopes, ECREF, Start Address and Control registers. The last column in the Address Map table lists the registers required for operating the High-Speed Buddy.

The Pixel and Delay 0 register arrays are not used in AMP mode. Therefore the user does not need to write to the Pixel or Delay 0 registers. In addition, the Amplitude 1, Amplitude 2, Delay 1, & Delay 2 register arrays are not used in AMP mode. Therefore, the user does not need to transfer values to these registers using the Data Transfer register. The Current DAC 0 & 1 registers are not used on High-Speed Buddy. The Analog Test and Digital Test registers are for test purposes and are not needed for operation but can be used for test. Although, the following registers are not needed for operation, they should be initialized to a known setting: Analog Test, Digital Test and Data Transfer.

5.3 Bus Cycles

A bus cycle to transfer data consists of a zip code being driven on the bus followed by one or more data words. Table 5-3 defines how EN, ALE and DCLK control the operations on the bus. A zip code is valid on the bus when EN and ALE are both high. When EN is high and ALE is low then data is on the bus. The enable signal (EN) essentially acts as a chip select so that the data lines and the ALE control signal can be bussed to multiple SLM212 driver chips. EN also operates as a "clock enable" and in this role determines the number of data words that are written into an array of registers. In addition, the enable signal allows the user to inject wait states in a burst of data by de-asserting EN. This feature will be useful in the event that the processor that sources the data cannot support a continuous data rate.

EN	ALE	DCLK	Operation
HI	HI	↑	Write Zip Code
HI	LOW	↑	Write Data
LOW	X	↑	No Operation
X	X	↓	No Operation

Table 5-3 Truth Table for Bus Operation

In the case of a single register the number of data words following the zip code is defined to be one. In the case of an array of registers, the start address is programmed into the "Start Address Register" and the EN signal controls the number of data words. The "Start Address Register" must be loaded in a bus cycle prior to the bus cycle for an array of registers. An internal address register is incremented or decremented automatically for each valid data write cycle. The Up/Down bit (D0) in the Control Register determines whether the internal address counter counts up or counts down.

Figure 5-1 shows a timing diagram of a bus cycle that writes to a single register. The "Start Address Register" is not used in this case since the bus cycle has only one data word. ALE and EN must be high to initiate the bus cycle and to indicate that a Zip Code is on the bus. EN is high and ALE is low when the data is valid on the bus. The high order 10-bits on the bus are not used.

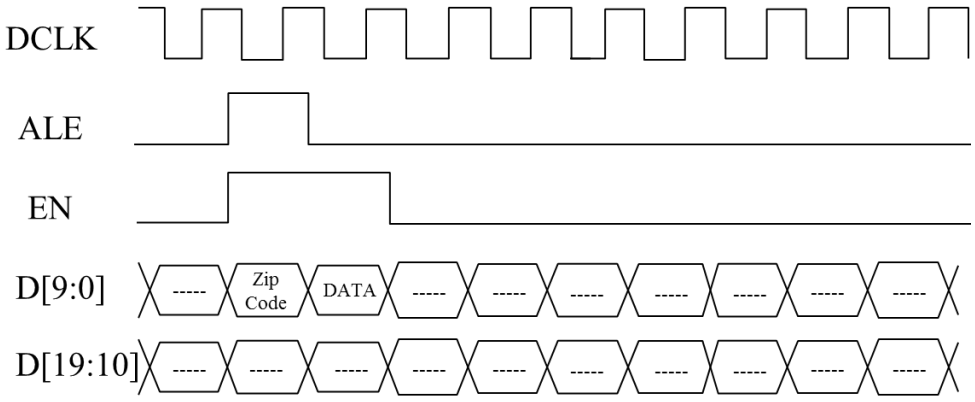

Figure 5-1 Bus Cycle: Write to a Single Register

Figure 5-2 shows a bus cycle that writes to an array of registers when in 20-bit bus mode and the address counter is counting up. The illustrated bus cycle writes 10 pixel amplitude values from driver channel first address of 0 to a last address of 9. EN and ALE must be high to initiate a bus cycle and to indicate a zip code is on the bus. The ALE signal (with EN high) causes the "Start Address Register" to be loaded into an internal address counter. Typically, the start address value would be "0", so then data D0 & D1 would be written to channels 0 & 1, respectively. EN is high and ALE is low to indicate that valid data is on the bus. The address counter continues to count and data continues to be sequentially written to the array of registers when EN is high.

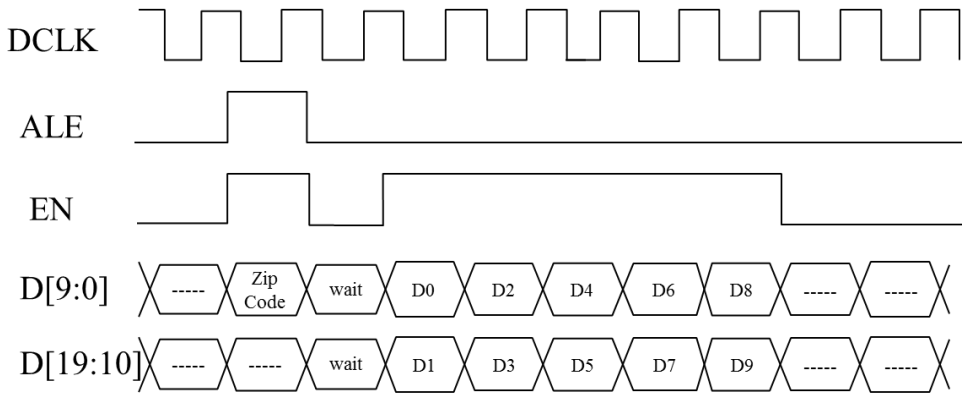

Figure 5-2 Bus Cycle: Write Pixel Amp Values (Counting Up) with Wait State

A wait state or no operation (NOP) is indicated when EN is low. A wait state is required prior to writing to channel pair (0,1) and channel pair (270, 271). The address counter only counts when EN is high. The bus cycle is terminated with the last cycle where EN is high or by the start of a new bus cycle (ALE high and EN high). The address counter can count up or count down.

The architecture that allows access to the Amplitude 0 Register imposes some operational requirements and restrictions since the Amplitude 0 Registers are always internally accessed as a channel pair consisting of an even and the adjacent higher odd channel. Therefore, when writing to the Amplitude 0 Register, external data must be written in channel pairs. In 20-bit bus mode, that means that an amplitude value must be present on both the lower 10-bits and on the upper 10-bits of the data bus.

When loading the Amplitude 0 Registers and counting up, the start address register must be loaded with an even channel address. A data pair is an even channel and the sequentially higher odd channel (for example channel 0 is paired with channel 1). When counting down in 20-bit mode, the start address register must also be loaded with the even channel address of the data pair. Figure 5-2 illustrates a valid bus cycle but additionally illustrates the particular requirements to write to the Amplitude 0 Register while counting up. The requirements and restrictions discussed in this paragraph only apply when writing to the Amplitude 0 Register and do not apply when writing to the other array registers (e.g. delay, pixel, slopes) since the upper 10-bits of the data bus are not used.

Figure 5-3 shows a bus cycle of a write to the Amplitude #0 array of registers in the 20-bit bus mode and when the address counter is counting down. The Start Address register should be preloaded with a value of 270 for writing to channels 271 to 262. Note that a wait state is required prior to channel pair (270, 271).

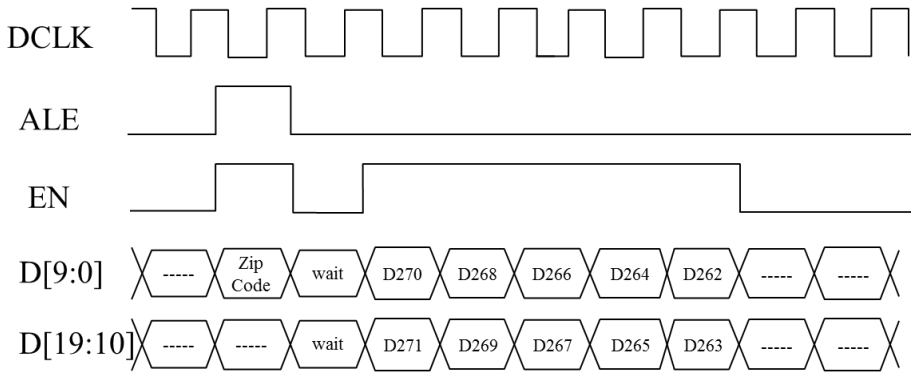

Figure 5-3 Bus Cycle: Write Pixel Amp Values (Counting Down) with Wait State

5.4 Data Register Formats

Individual registers within register arrays are addressed by using the address counter. The address counter is initialized from the start address register. The address counter can count up or down

Amplitude 0 Registers:

- Channel Amplitude #0 data is placed on the 20-bit data bus as follows:
  - D[19:10]: Odd channel data
  - D[9:0]: Even channel data
- The register address range spans 0..271 for driver channels 0..271.

Data Bus										Register Address	Channel Number
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	0	0
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	1	1
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	2	2
…	…	…	…	…	…	…	…	…	…	…	…
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	271	271
Even or Odd Channels											

Table 5-4 Amplitude 0 Register Array Map in 10-bit Mode

The Amplitude 0 registers are used during the two operational modes – PWM mode and AMP mode. The Amplitude 1 and Amplitude 2 registers are loaded by the user only during PWM mode. However, some of the internal array registers may be used by the control circuitry.

Pixel Registers:

- 8 one-bit pixel values [P7:P0] are assigned from the least significant bits of the data bus [D7:D0].
- The register address range spans 0..33 for driver channels 0..271 in groups of 8.
- For example, D7 supplies pixel bit values for driver channels 7, 15, 23, … 271.
- For example, D0 supplies pixel bit values for driver channels 0, 8, 16, …264.
- Pixel registers are only used during the PWM mode of operation.

Data Bus										Register Address	Channel Number
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
		C7	C6	C5	C4	C3	C2	C1	C0	0	7-0
		C15	C14	C13	C12	C11	C10	C9	C8	1	15-8
		C23	C22	C21	C20	C19	C18	C17	C16	2	23-16
		…	…	…	…	…	…	…	…	…	…
		C271	C270	C269	C268	C267	C266	C265	C264	33	271-264

Table 5-5 Pixel Array Register Map

Delay 0 Registers:

- 8 delay value bits [V7:V0] are assigned from the least significant bits of the 10-bit data bus [D7:D0].
- The register address range spans 0..271 for driver channels 0..271.
- Delay 0 registers are only used in PWM & "AMP mode with Delay" modes of operation

Data Bus										Register Address	Channel Number
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
		V7	V6	V5	V4	V3	V2	V1	V0	0	0
		V7	V6	V5	V4	V3	V2	V1	V0	1	1
		V7	V6	V5	V4	V3	V2	V1	V0	2	2
		…	…	…	…	…	…	…	…	…	…
		V7	V6	V5	V4	V3	V2	V1	V0	271	271

Table 5-6 Delay 0 Array Register Map

The Delay 1 and Delay 2 registers are loaded by the user during PWM mode only.

The array registers Delay 1 and Delay 2 are not addressed directly using a zip code and starting address. These registers are written by copying from the Delay 0 registers according to bits in the Data Transfer register.

Slope Registers:

- 10 slope bits [V9:V0] are assigned from the data bus bits [D9:D0].
- The register address range spans 0..15 for 16 driver slope segments.
- The 16 driver slope segments apply to all of the 272 driver channels, the Current DAC0 channel, and the Current DAC1 channel.

Data Bus										Register Address	Slope Number
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0		
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	0	0
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	1	1
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	2	2
…	…	…	…	…	…	…	…	…	…	…	…
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	15	15

Table 5-7 Slopes Array Register Map

Programmable ECREF Register:

- 10 ECREF bits [V9:V0] are assigned from the data bus bits [D9:D0].
- This register is a single register.
- The Programmable ECREF setting applies to all 272 driver channels.

Data Bus										Register Address
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	N/A

Table 5-8 Programmable ECREF (Single) Register Map

Current DAC 0 Register:

- 10 Current DAC 0 value bits [V9:V0] are assigned from the 10-bit data bus [D9:D0].
- This register is a single register.

Data Bus										Register Address
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	N/A

Table 5-9 Current DAC 0 (Single) Register Map

Current DAC 1 Register:

- 10 Current DAC 1 value bits [V9:V0] are assigned from the 10-bit data bus [D9:D0].
- This register is a single register.

Data Bus										Register Address
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
V9	V8	V7	V6	V5	V4	V3	V2	V1	V0	N/A

Table 5-10 Current DAC 1 (Single) Register Map

Start Address Register:

- 9 address bits [A8:A0] are assigned from the data bus bits [D8:D0] to span the maximum register array range 0:271.
- The lowest nine bits of this register are multiplexed to the digital test circuit per the Digital Test register setting for the input mux address.
- This register is a single register.

Data Bus										Register Address
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
	V8	V7	V6	V5	V4	V3	V2	V1	V0	N/A

Table 5-11 Start Address (Single) Register Map

Control Register:

- 10 control bits are assigned to the data bus bits [D9:D0].
- This register is a single register.

Data Bits	Name	Description
D9	NoDelay	1 for AMP no-delay mode (for update per COL Signal); 0 for AMP with Delay mode and PWM mode (for update per WCLK signal).
D8	PwrMode	1 selects constant power mode; 0 selects variable power mode.
D7	PDn	1 for power down of test circuit.
D6	Tgain	Gain select for test circuit, –1 selects 75% of the full gain; 0 selects full gain.
D5	TRST	Reset test circuit, positive pulse to reset, width>10us, period <=1ms.
D4	Mode 10-bit	1 for 10 bit mode, 0 for 20 bit mode (select 20 bit mode for HS-Buddy)
D3	En-TOUTA	Enables the tri-state driver for the Analog Test Output Pin (TOUTA)
D2	En-TOUTD	Enables the tri-state driver for the Digital Test Output Pin (TOUTD)
D1	PWM-En	1 for PWM mode; 0 for AMP mode.
D0	Up/Down	The internal address counter counts up when high & counts down when low.

Table 5-12 Control (Single) Register Map

Data Transfer Register:

- 10 control bits are assigned to the data bus bits [D9:D0].
- This register is a single register.
- The Data Transfer Register bits D9, D8, & D7 control the parallel parity circuit.
- The Data Transfer Register bit D6 selects the source of the TOUTD digital test output pin – either parallel parity output or the digital test block output that includes the serial parity output.
- The Data Transfer Reg. bits D5:D0 control transfers to Amp 1 & 2, Delay 1 & 2 register arrays.

Data Bits	Name	Description
D9	ParityEN	1= Enables parity checking.
D8	ParityCLR	1= Clears Parity Register; ParityCLR is active only when ParityEN is 0.
D7	ParityEVEN	1 = Expect EVEN parity and output error if odd; 0 = Expect ODD parity and output error if even.
D6	DT SEL	Select output on DTST line: 0 = Parallel Parity 1 = Digital Test Block (includes serial parity)
D5	PWM2	1=Transfer Data From PWM Latch 0 to PWM Latch 2.
D4	PWM1	1=Transfer Data From PWM Latch 0 to PWM Latch 1.
D3	AMP2	1=Transfer Data From Amplitude Latch 0 to Amplitude Latch 2.
D2	AMP1	1=Transfer Data From Amplitude Latch 0 to Amplitude Latch 1.
D1	GATE	1= Enable write & read operations for the Amplitude and Delay Latches
D0	P_EN	1= Enable the output of the currently selected Delay Latch

Table 5-13 Data Transfer (Single) Register Map

Analog Test Register:

- 10 control bits are assigned to the data bus bits [D9:D0].
- This register is a single register.

Data Bits	Name	Description
D9	TOUT	1 Selects TSO; 0 selects THVO.
D8:D7	TS1:TS0	Selects which of TSO/THVO [3:0] to output.
D6:D0	A6:A0	Test channel address, 0-69 (each address selects 4 channels).

Table 5-14 Analog Test (Single) Register Map

Digital Test Register:

- 10 control bits are assigned to the data bus bits [D9:D0].
- This register is a single register.
- The Digital Test Register bits D9:D6 control the serial parity output.
- The Digital Test Register bits D5:D0 select the serial parity input; when D9 is zero, serial parity is bypassed and the digital test block output is the same as the serial parity input.
- The Digital Test Register bits D5:D0 select the serial parity input from 64 choices. Note the choices are detailed in the section on parity.

Data Bits	Name	Description
D9	DTPAROUT	1 selects digital test parity output; 0 bypasses digital test (serial) parity circuit.
D8	DTODDEVN	1 Expect EVEN parity and output error if odd; 0 Expect ODD parity and output error if even.
D7	DTCLRBAR	0 clears the digital test (serial) parity flip/flop.
D6	DTPAREN	Digital test parity enable – 1 enables parity checking in the digital test circuit; 0 retains previous parity.
D5:D0	A5:A0	Address to input mux.

Table 5-15 Digital Test (Single) Register Map

5.5 Operation of the Data Transfer Register

Operation of the Data Transfer register is not required for AMP mode, therefore, is does not need to be operated for High-Speed Buddy.

5.6 Driver Timing

Figure 5-4 and Table 5-16 show the timing specifications for WCLK and DCLK. Table 5-17 and Figure 5-5 show the timing specifications for the data interface.

The user is responsible for ensuring that all pixel data transmission is completed between COL strobes. The last word of the pixel data must be clocked into the driver prior to asserting the COL signal. This timing relationship is defined below.

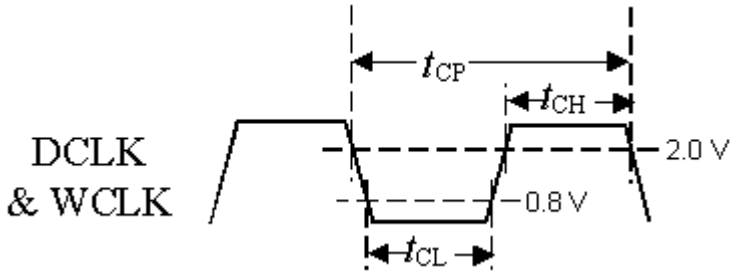

Figure 5-4 Timing Specifications for Clocks

WCLK				
Paramete	Description	Min	Ma	Unit
t_{CPW}	WCLK cycle time	7.		nse
t_{CHW}	WCLK high	2.6		nse
t_{CLW}	WCLK low	2.6		nse
DCLK				
Paramete	Description	Min	Ma	Unit
t_{CPD}	DCLK cycle time	7.		nse
t_{CHD}	DCLK high	2.6		nse
t_{CLD}	DCLK low	2.6		nse

Table 5-16 Timing Specifications for Clocks

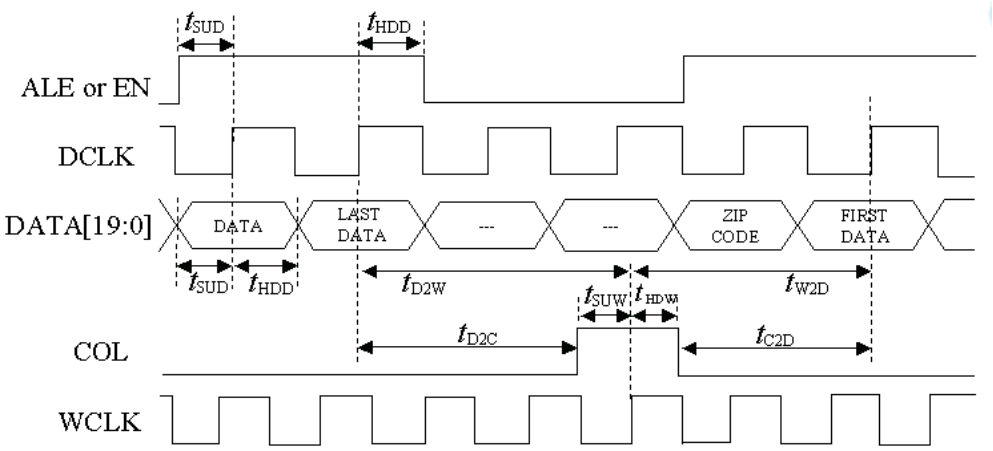

Figure 5-5 Timing Specifications for Data Interface

Parameter	Description	Min	Max	Units
	All Modes			
t_{SUD}	Set up time to rising edge of DCLK	2		nsec
t_{HDD}	Hold time from rising edge of DCLK	0		nsec
	PWM Mode & Calibration Mode with Delay			
t_{SUW}	Set up time to rising edge of WCLK	2		nsec
t_{HDW}	Hold time from rising edge of WCLK	0		nsec
t_{D2W}	Rising edge of DCLK to rising edge of WCLK (1)	$4t_{CPD} - 2t_{CPW}$		nsec
t_{W2D}	Rising edge of WCLK to rising edge of DCLK (2)	$4t_{CPW} - 2t_{CPD}$		nsec
	Calibration Mode			
t_{D2C}	Rising edge of DCLK to rising edge of COL (3)	$4t_{CPD}$		nsec
t_{C2D}	Falling edge of COL to rising edge of DCLK (4)	0		nsec

1. Time from DCLK of the last pixel data of current column to first WCLK where COL is asserted
2. Time from last WCLK where COL is asserted to DCLK of the first pixel data of next column
3. Time from DCLK of the last pixel data of current column to rising edge of COL
4. Time from falling edge of COL to DCLK of the first pixel data of next column

Table 5-17 Timing Specifications for Data Interface

6 Appendix C – SLM212 Driver: DACs and Analog References

6.1 Simplified Model Circuit for HV DACs

The SLM212 analog section is a current based design. For each of the 272 channels, a 10-bit programmable current source sinks current from the high voltage VDDAH supply through a load resistor. The corresponding drop in voltage across the load resistor creates an adjustable voltage on the individual channel HVout pad which is connected between the load resistor and the programmable current source. This "pull-down" configuration results in a high voltage analog output with a 10-bit resolution. A simplified schematic of the pull-down architecture is shown in Figure 6-1. A high voltage buffer HVFET is used to isolate standard 3.3 V CMOS circuitry from the high voltage VDDAH supply.

Internally, each 10-bit DAC is divided into sixteen 6-bit sections. The voltage drop per bit, or slope, of each of these 16 segments can be individually adjusted with 10-bits of accuracy. An additional gain adjustment, ECRef, has been designed into the chip that controls the maximum slope that any single segment can achieve. Thus the 1024 output levels of each DAC are divided into 16 slope sections of 6 bits each. While each of the 16 slope segments can be adjusted individually, all 272 channels on the chip share the corresponding slope values.

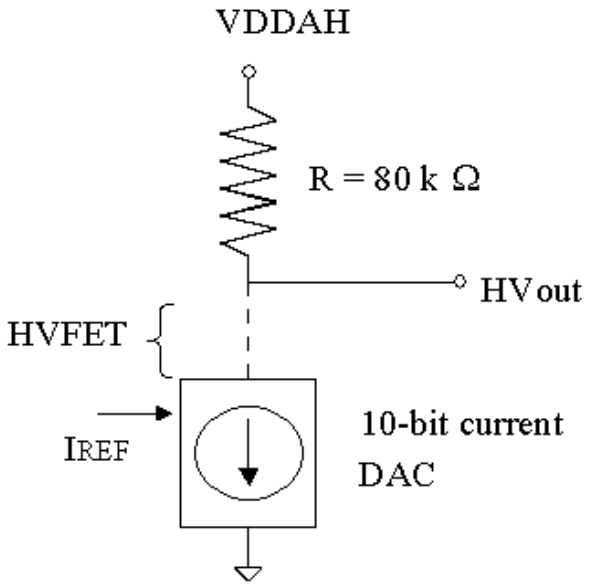

Figure 6-1 Simplified schematic of SLM212 pull-down design

6.2 Global Reference – ECRef

A 10-bit register located at zip code 200h controls the global reference ECRef. ECRef controls the maximum slope that any of the 16 segments can achieve. At the nominal value of 512 for ECRef, a single slope segment (or 64 DAC levels) will cover ~25% of the VDDAH range when its slope is set to 255. The ECRef register allows this maximum slope to be adjusted +/-25%.

To optimize driver performance for variable applied VDDAH, the internal circuitry references the maximum slope value produced by ECRef to the applied VDDAH and the load resistor as shown in Figure 6-2. This means that regardless of the actual VDDAH applied, a single segment with slope=255 will still cover ~25% of the VDDAH range when ECRef is 512.

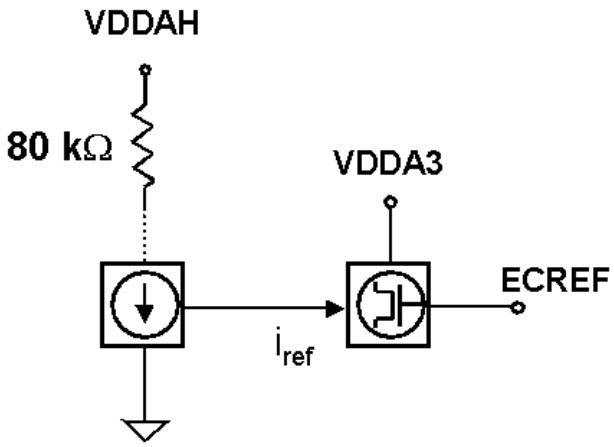

Figure 6-2 ECREF generating circuitry with VDDAH compensation

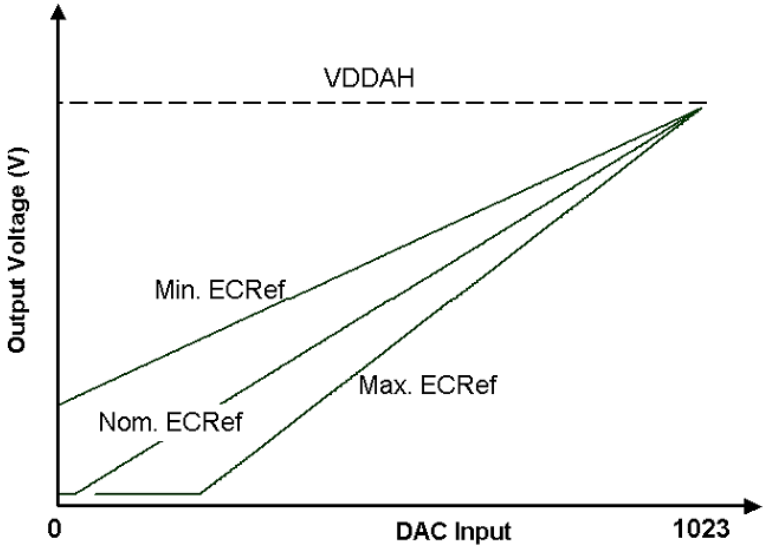

Figure 6-3 SLM212 high voltage output as a function of DAC code for varying ECREF

6.3 Reference generation for slope segments

As discussed above, the 10-bit DAC for each channel is divided into sixteen 6-bit segments that sum together to form the 10-bit current source. Each of these segments can be configured with its own output slope. ECRef controls the range of possible slopes for each segment. An array of 16 registers located at ZIP Code 007h controls these slope generators. Each generator has a range from 0 to the maximum value programmed by ECRef and is configurable within this range with a 10-bit resolution. (It is worth noting that because these slopes are derived from the ECRef value and ECRef is derived from VDDAH, the slope generators will automatically adjust to different values for VDDAH as well). All channels share the slope generators, so all channels will have the same DAC vs. Voltage output curve.

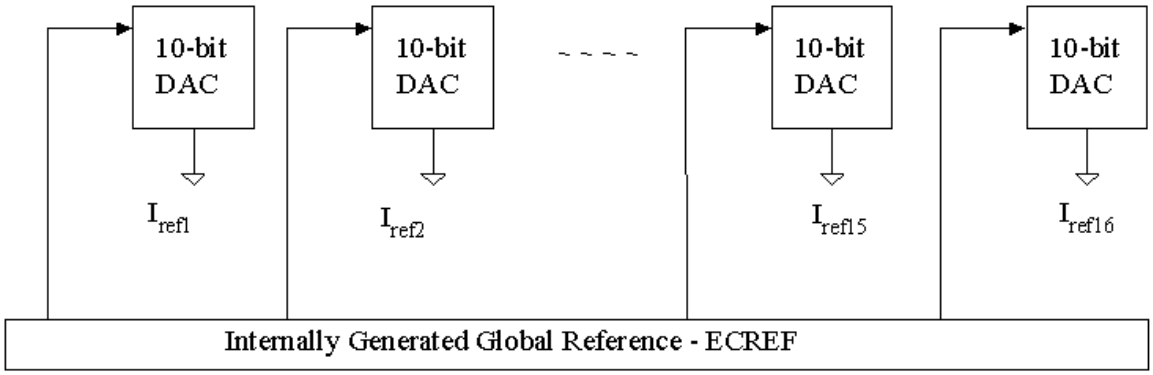

Figure 6-4 Simplified Schematic Of SLM212 Current Generating Architecture

6.4 Slope, Amplitude and Reference Polarity

In the SLM212 pull-down design, maximum output voltage is always achieved at DAC code 1023, and is equal to VDDAH-1LSB (A minimum of 1LSB of current is always drawn through the load resistor to ensure stability of the output voltage). As the DAC code for the channel is reduced, the output voltage is also reduced in accordance with the Slope settings for each segment. Across each slope segment the output voltage is linear with respect to DAC code, in other words the voltage steps are equal. This is not necessarily true across slope segments where the programmed slope can vary. For nominal operation, where the full operational range is used with an arbitrary slope profile, the sum of all slope segments over the 1023 DAC levels should equal 100% of VDDAH. An example of output voltage as a function of DAC code with arbitrary slope settings is shown in Figure 6-5. At each slope segment, the digital code of the slope is given.

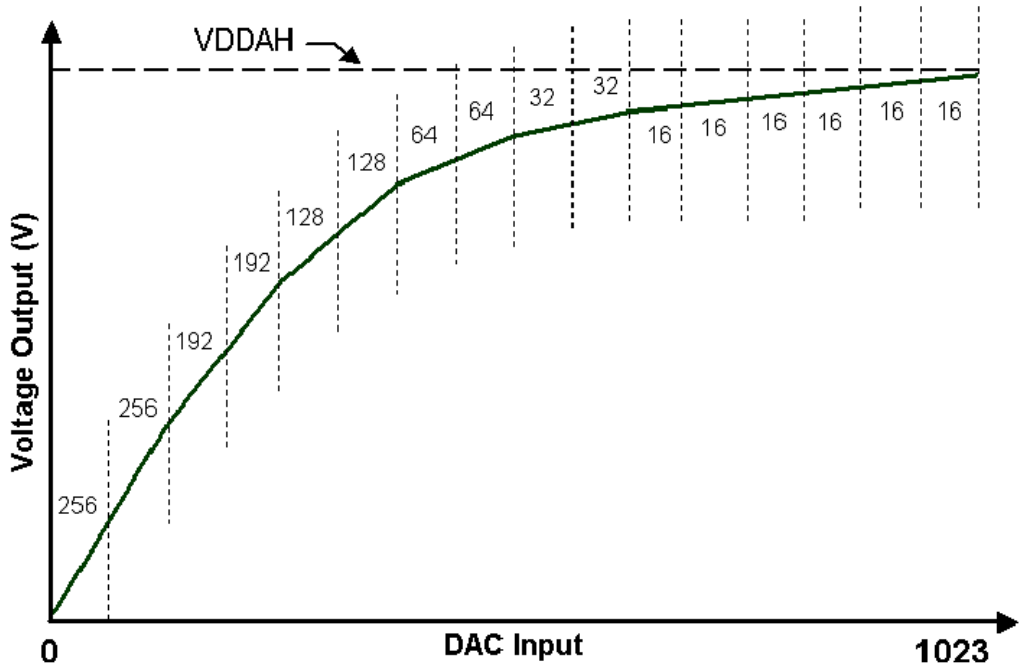

Figure 6-5 Voltage output versus DAC code for arbitrary slope settings

A high slope setting causes a rapid decrease in voltage. If slopes are set to high values for several segments, it is possible to reach the minimum voltage output of the channel before reaching DAC code 0. For example, if the ECRef is set to nominal (max of 25% per segment) and the first 5 segments have slopes set to 1023 (max slope), then the minimum output voltage will be reached around DAC code 768 (4 segments). For all DAC codes below 768 the voltage would remain at the minimum.

A low slope setting causes a slow decrease in voltage from DAC code 1023. In this case, it is possible that the minimum achievable voltage will not be reached and even at DAC code 0. If all slope values are set to 0, at DAC code 0, the voltage output would be very close to VDDAH (there is always a very slight minimum slope to each segment).

6.5 Description of Output Characteristics:

The SLM212 can be programmed to achieve full pull-down from VDDAH within two slope sections, thus at DAC code 1023-2*64=895. For this to happen, ECRef should be set to 512 and the slope settings should be 512. A faster pull-down, or steeper slope, can be achieved with higher slope DAC value, up to slope 1023. However, simulation shows that, for this setting, the glitch energy associated with segment transitions could be higher than the specified 10nVs, and is therefore not recommended or guaranteed. Segment transition glitches occur due to switching between banks of transistors and are specified for HVFET transistor gate voltage VggHV higher than 3.1 V.

The SLM212 is specified for operation between 1 and 22 V analog output, although simulations predict a minimum voltage of around 0.8 V. The predicted Integral Non-Linearity from 1 – 22 V is less than 1.2 LSB.

Table 6-1 shows the current per bit, current per slope segment (64 bits), and pull-down voltage for a nominal load resistor of 80 kOhm. Nominal operating conditions are for ECREF setting of 512, which result in 20 mV/bit at slope 64.

Digital Input D[9:0]	Slope Per bit	Slope Per Segment	Pull-down Voltage/Segment (@ 80 kOhm)	Comments
1023	4 µA	256 µA	20.48	Spec Not Guaranteed
512	2 µA	128 µA	10.24	Max. Slope
256	1 µA	64 µA	5.12	
128	0.5 µA	32 µA	2.56	
64	0.25 µA	16 µA	1.28	Standard Slope 20 mV/Bit
32	0.125 µA	8 µA	0.64	
16	62.5 nA	4 µA	0.32	
8	31.25 nA	2 µA	0.16	
4	15.6 mA	1 µA	0.08	
2	7.81 nA	0.5 µA	0.04	0.63 mV/Bit

Table 6-1 Current and Voltage Characteristics of SLM212 ECREF/Slope design

7 Appendix D – I²C Basics

The I²C specification defines how data is transferred on the I²C bus. The data packet starts with a control byte (i.e. start byte), which is composed of a 7-bits slave address and the least-significant bit is a Read/Write bit. Any data that follows is user defined. Figure 7-1 illustrates a write data transfer where the I²C master writes data to a slave. Figure 7-2 illustrates a read data transfer where the I²C master reads data from a slave.

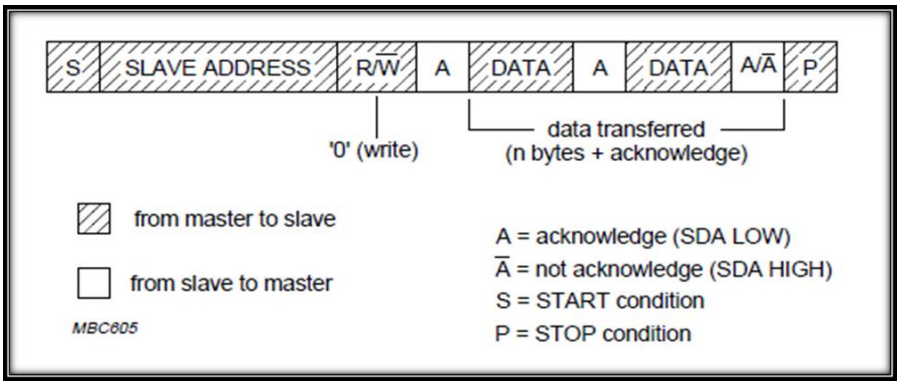

Figure 7-1 I²C WRITE: Master Writes Data to Slave

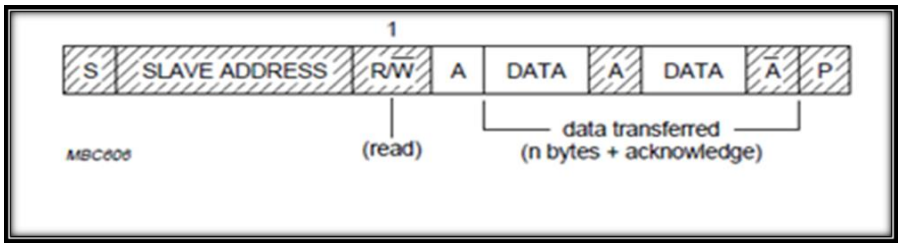

Figure 7-2 I²C READ: Master Reads Data from Slave

7.1 Sending Data from I²C Bus Master to Slave

Step 1: The I²C bus master (on the host controller) begins data transfer by generating a START condition, this brings the SDA line from high to low while the SCL line is high.

Step 2: The bus master transmits the control byte plus the user data bytes (see Table 7-1) onto the I²C bus as following:

<control-byte> <user-byte-1> <user-byte-2> … <user-byte-M>

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x00)	Its upper 7 bits are the lower 7 bits of the I²C slave address; its LSB is 0 for write operation.
User Bytes	0 to 255	These are all the user bytes sent from master to slave

Table 7-1 Control (Address) Byte and User Bytes to be Sent from Master to Slave

After transmitting each byte to the slave device, the I^2C controller that acts as the bus master shall receive an acknowledgement from the slave device before sending out the next byte. If no acknowledgement is received from the slave device after a byte transmission, the I^2C controller shall abort the data transmission (see Step 3). However, if the I^2C controller fails to get an acknowledgement from the slave (PSoC) after the Control (Address) Byte transmission on the first attempt, the controller shall stop the data transmission and wait 20 milliseconds or more, then try to re-transmit the Control Byte plus the user data bytes (abort the data transmission if it still fails).

Step 3: The bus master stops data transmission by generating a STOP condition, this brings the SDA line from low to high while the SCL line is high.

7.2 Receiving Data from Slave by I^2C Bus Master

Step 1: The I^2C bus master starts data transfer by generating a START condition, this brings the SDA line from high to low while the SCL line is high.

Step 2: The bus master transmits the control byte (see Table 7-2) onto the I^2C bus:

<control-byte>

Byte Type	Value	Description
Control Byte	((0x29<<1) | 0x01)	Its upper 7 bits are the lower 7 bits of the I^2C slave address; its LSB is 1 for read operation.

Table 7-2 Control (Address) Byte to be Sent before Receiving Data from Slave

Step 3: After transmitting the Control (Address) Byte, the I^2C controller that acts as the bus master shall receive an acknowledgement from the targeted slave device (PSoC). If the I^2C controller fails to get an acknowledgement on the first attempt, the controller shall stop the data transfer (see Section 7.1 Step 3) and wait 20 milliseconds or more, then try to re-transmit the Control Byte and proceed with the data read operation (abort the data transfer if it still fails).

Step 4: After the Control Byte transmission is acknowledged by the slave (PSoC), the I^2C bus master starts requesting data bytes from the slave by switching to the "read-from-slave" mode.

Step 5: If the bus master is requesting only one (1) data byte from the slave, then it also tells the slave (PSoC) to send only one (1) data byte. One dummy-byte read may be needed (to initiate the first byte receiving) before collecting the actual data byte. The I^2C bus master reads one (1) data byte from the slave device.

If the I^2C bus master is requesting more than one (1) data byte from the slave device, then it tells the slave device to send more data when requesting a data byte that is not the last one. One dummy-byte read may be needed (to initiate the first byte receiving) before collecting the actual data bytes. The I^2C bus master reads the data bytes from the slave device one-byte at a time until it receives all the requested data bytes. Before receiving the last data byte, the bus master must inform the slave device to stop putting more data bytes on the I^2C bus after this "last" byte.

Step 6: The bus master stops the data read operation by generating a STOP condition, this brings the SDA line from low to high while the SCL line is high.

8 Appendix E - EEPROM Data Specifications

8.1 Introduction

The High-Speed Buddy GLV module uses SLM212 driver chips with 10-bit DAC's. The Cricket Interface Board uses a PSoC with a 8-bit micro-controller unit (Programmable-System-on-Chip, CY8C27543, 44-pin TQFP, Cypress Microsystems) to perform Built-In-Test (BIT) functions (A/D conversions), key control voltages applied to the GLV (D/A conversions), and provide 4096 bytes of non-volatile memory (EEPROM) for storing GLV configuration and initialization data as well as user-defined data.

The read and write access to this PSoC EEPROM is via the I^2C interface to the PSoC. The 4kBytes of PSoC EEPROM is organized into 16 memory banks of 256 bytes each. Data can be read from each bank at any offset within the bank (0 to 255) and size of the data to be read can range from 1 to (256-offset) bytes. Data can be written to each of the first 8 memory banks at one of these four valid offset values (0, 64, 128, and 192) within the bank, and the number of bytes to be written to the EEPROM during a single write-sequence must be exactly 64 bytes. These first 8 memory banks are allocated to user to store user-defined data.

The second half of the PSoC EEPROM area (the next 8 memory banks) is used by SLM to store information such as, the part numbers and revision levels of GLV die and driver dies, serial number and part number of the module board, specifications of the GLV, upper and lower limits and the recommended default DAC settings for various voltages or currents (VDDAH, ECRef, Bias, Common and Slopes) to be applied to the GLV, and parameters used to calibrate the A/D and D/A converter channels on the PSoC. User is not allowed to write to these 8 EEPROM banks; data can only be written to these EEPROM banks by SLM at the factory or during the PSoC firmware upgrade.

I^2C command protocols are provided for the user to store or retrieve data to or from the EEPROM (see Sections 3.2.3.4 and 3.2.3.5 for details). Following sections specify the data format for each of the sixteen EEPROM banks.

Note that all the 16-bit (2-byte) and 32-bit (4-byte) integers (signed and unsigned) defined in the following sections are read from or written to the EEPROM in BIG-ENDIAN data format!

8.2 Serial EEPROM Banks #0 to #7

To be defined by user (total size is 8 banks x 256 bytes/bank = 2048 bytes). User has Read-and-Write access to these EEPROM banks.

8.3 Serial EEPROM Bank #8

User has Read-Only access to this EEPROM bank.

Entry	Range	Data Format (16 bytes)	Notes
1	0x00-0x0F	0x00-0x03 Compatibility code 0x04-0x04 EEPROM version number (major) 0x05-0x05 EEPROM version number (minor) 0x06-0x06 Firmware version number (major) 0x07-0x07 Firmware version number (minor) 0x08-0x09 Firmware version number (bug-fix id) 0x0A-0x0B Frame delay in μS for test patterns 0x0C-0x0D Column time in μS for test patterns 0x0E-0x0F number of GLV channels per driver	Compatibility code is "HSB1". EEPROM and firmware version numbers are used to track EEPROM data and firmware changes. EEPROM version number (major) shall be >= 2. Frame delay and column time are 16-bit unsigned integers.
2	0x10-0x1F	0x10-0x10 number of GLV drivers 0x11-0x11 number of GLV drive-level bits 0x12-0x13 default Amplitude0 setting 0x14-0x15 ZIP code for setting Amplitude0 0x16-0x17 ZIP code for setting single-bit Pixels 0x18-0x19 default Delay0 setting 0x1A-0x1B maximum Delay0 setting 0x1C-0x1D ZIP code for setting Delay0 0x1E-0x1F default Start-Address for driver 0	# of drive-level bits defines max drive-level (e.g., 10-bits=>1023). Amplitude0, Delay0 and Start-Address settings are two-byte unsigned integers, their minimum settings shall be 0. Maximum Amplitude0 value is set by the # of drive-level bits.
3	0x20-0x2F	0x20-0x21 default Start-Address for driver 1 0x22-0x23 default Start-Address for driver 2 0x24-0x25 default Start-Address for driver 3 0x26-0x27 ZIP code for Start-Address Register 0x28-0x29 default Control Register for driver 0 0x2A-0x2B default Control Register for driver 1 0x2C-0x2D default Control Register for driver 2 0x2E-0x2F default Control Register for driver 3	Maximum Start-Address is set by # of GLV channels per driver. Default Amplitude0 and Delay0 register values defined here (at 0x12 and 0x18 in bank #8) are for all drivers on the GLV module as well as for all driver channels.
4	0x30-0x3F	0x30-0x31 ZIP code for (driver) Control Register 0x32-0x33 default DataTransfer Reg for driver 0 0x34-0x35 default DataTransfer Reg for driver 1 0x36-0x37 default DataTransfer Reg for driver 2 0x38-0x39 default DataTransfer Reg for driver 3 0x3A-0x3B ZIP code for Data-Transfer Register 0x3C-0x3D default Digital-Test Reg for driver 0 0x3E-0x3F default Digital-Test Reg for driver 1	
5	0x40-0x4F	0x40-0x41 default Digital-Test Reg for driver 2 0x42-0x43 default Digital-Test Reg for driver 3 0x44-0x45 ZIP code for Digital-Test Register 0x46-0x47 default Analog-Test Reg for driver 0 0x48-0x49 default Analog-Test Reg for driver 1 0x4A-0x4B default Analog-Test Reg for driver 2 0x4C-0x4D default Analog-Test Reg for driver 3 0x4E-0x4F ZIP code for Analog-Test Register	
6	0x50-0x5F	GLV module type string (including driver type): AA…AA-NNN (must end with a null byte) (e.g., "HIGH-SPEED BUDDY-212\0").	A – character (A-Z) for module name N – digit (0-9) for driver type
7	0x60-0x6F	0x60-0x69 Manufacture date: YYYY-MM-DD 0x6A-0x6F Module Serial Number: NNNNNN	Date example: 2016-01-01 N – digit (0-9) for serial number
8	0x70-0x7F	GLV Die part number and revision level: ICA-NNNNNN-NN-RR	N – digit (0-9) R – revision level

Entry	Range	Data Format (16 bytes)	Notes
9	0x80- 0x8F	GLV Die ID (string shall end with a null byte): NNNNNN-NN-AN	A – character (A-Z) N – digit (0-9)
10	0x90- 0x9F	Driver Die part number and revision level: ICX-NNNNNN-NN-RR	N – digit (0-9) R – revision level
11	0xA0- 0xAF	Driver-0 (U1) Die ID (shall end with a null byte): ANNNNN-NNAN-AN	A – character (A-Z) N – digit (0-9)
12	0xB0- 0xBF	Driver-1 (U2) Die ID (shall end with a null byte): ANNNNN-NNAN-AN	A – character (A-Z) N – digit (0-9)
13	0xC0- 0xCF	Driver-2 (U3) Die ID (shall end with a null byte): ANNNNN-NNAN-AN	A – character (A-Z) N – digit (0-9)
14	0xD0- 0xDF	Driver-3 (U4) Die ID (shall end with a null byte): ANNNNN-NNAN-AN	A – character (A-Z) N – digit (0-9)
15	0xE0- 0xEF	GLV module PCB part number and rev. level: PWA-NNNNNN-NN-RR	N – digit (0-9) R – revision level
16	0xF0- 0xFF	GLV module PCB serial number: ANNNN-NNNNN-NNN	A – character (A-Z) N – digit (0-9)

Table 8-1 Serial EEPROM Bank #8

8.4 Serial EEPROM Bank #9

User has Read-Only access to this EEPROM bank.

Entry	Range	Data Format (16 bytes)	Notes
1	0x00-0x0F ADC channel 0	0x00-0x03 minimum analog value 0x04-0x07 maximum analog value 0x08-0x09 minimum digital data 0x0A-0x0B maximum digital data 0x0C-0x0F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].[1]
2	0x10-0x1F ADC channel 1	0x10-0x13 minimum analog value 0x14-0x17 maximum analog value 0x18-0x19 minimum digital data 0x1A-0x1B maximum digital data 0x1C-0x1F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
3	0x20-0x2F ADC channel 2	0x20-0x23 minimum analog value 0x24-0x27 maximum analog value 0x28-0x29 minimum digital data 0x2A-0x2B maximum digital data 0x2C-0x2F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
4	0x30-0x3F ADC channel 3	0x30-0x33 minimum analog value 0x34-0x37 maximum analog value 0x38-0x39 minimum digital data 0x3A-0x3B maximum digital data 0x3C-0x3F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
5	0x40-0x4F ADC channel 4	0x40-0x43 minimum analog value 0x44-0x47 maximum analog value 0x48-0x49 minimum digital data 0x4A-0x4B maximum digital data 0x4C-0x4F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
6	0x50-0x5F ADC channel 5	0x50-0x53 minimum analog value 0x54-0x57 maximum analog value 0x58-0x59 minimum digital data 0x5A-0x5B maximum digital data 0x5C-0x5F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
7	0x60-0x6F ADC channel 6	0x60-0x63 minimum analog value 0x64-0x67 maximum analog value 0x68-0x69 minimum digital data 0x6A-0x6B maximum digital data 0x6C-0x6F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
8	0x70-0x7F ADC channel 7	0x70-0x73 minimum analog value 0x74-0x77 maximum analog value 0x78-0x79 minimum digital data 0x7A-0x7B maximum digital data 0x7C-0x7F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].

[1] The units for ADC channels’ analog floating-point data, as well as the channel names, are defined in Section 3.2.3.7 Table 3-22.

Entry	Range	Data Format (16 bytes)	Notes
9	0x80-0x8F ADC channel 8	0x80-0x83 minimum analog value 0x84-0x87 maximum analog value 0x88-0x89 minimum digital data 0x8A-0x8B maximum digital data 0x8C-0x8F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
10	0x90-0x9F ADC channel 9	0x90-0x93 minimum analog value 0x94-0x97 maximum analog value 0x98-0x99 minimum digital data 0x9A-0x9B maximum digital data 0x9C-0x9F scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
11	0xA0-0xAF ADC channel 10	0xA0-0xA3 minimum analog value 0xA4-0xA7 maximum analog value 0xA8-0xA9 minimum digital data 0xAA-0xAB maximum digital data 0xAC-0xAF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
12	0xB0-0xBF ADC channel 11	0xB0-0xB3 minimum analog value 0xB4-0xB7 maximum analog value 0xB8-0xB9 minimum digital data 0xBA-0xBB maximum digital data 0xBC-0xBF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
13	0xC0-0xCF ADC channel 12	0xC0-0xC3 minimum analog value 0xC4-0xC7 maximum analog value 0xC8-0xC9 minimum digital data 0xCA-0xCB maximum digital data 0xCC-0xCF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
14	0xD0-0xDF ADC channel 13	0xD0-0xD3 minimum analog value 0xD4-0xD7 maximum analog value 0xD8-0xD9 minimum digital data 0xDA-0xDB maximum digital data 0xDC-0xDF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
15	0xE0-0xEF ADC channel 14	0xE0-0xE3 minimum analog value 0xE4-0xE7 maximum analog value 0xE8-0xE9 minimum digital data 0xEA-0xEB maximum digital data 0xEC-0xEF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].
16	0xF0-0xFF ADC channel 15	0xF0-0xF3 minimum analog value 0xF4-0xF7 maximum analog value 0xF8-0xF9 minimum digital data 0xFA-0xFB maximum digital data 0xFC-0xFF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].

Table 8-2 Serial EEPROM Bank #9

8.5 Serial EEPROM Bank #10

User has Read-Only access to this EEPROM bank.

Entry	Range	Data Format (16 bytes)	Notes
1	0x00-0x0F	0x00-0x01 default VDDAH digital setting 0x02-0x03 VDDAH error level (digital data) 0x04-0x05 minimum VDDAH digital setting 0x06-0x07 maximum VDDAH digital setting 0x08-0x09 minimum VDDAH in milli-volts (analog) 0x0A-0x0B maximum VDDAH in milli-volts (analog) 0x0C-0x0C PSoC DAC channel to set VDDAH 0x0D-0x0D PSoC ADC channel to read VDDAH 0x0E-0x0F scale to convert ADC analog to mV	All VDDAH values are two-byte integers. Scale factor is two-bye unsigned integer that scales the ADC analog value (signed long) defined in bank #9 down to VDDAH in milli-volts.
2	0x10-0x1F	0x10-0x11 default Common digital data 0x12-0x13 Common error level (digital data) 0x14-0x15 minimum Common digital setting 0x16-0x17 maximum Common digital setting 0x18-0x19 minimum Common in millivolts (analog) 0x1A-0x1B maximum Common in millivolts (analog) 0x1C-0x1C PSoC DAC channel to set Common 0x1D-0x1D PSoC ADC channel to read Common 0x1E-0x1F scale to convert ADC analog to mV	All Common values are two-byte integers. Scale factor is two-bye unsigned integer that scales the ADC analog value (signed long) defined in bank #9 down to Common in milli-volts.
3	0x20-0x2F	0x20-0x21 default Bias digital setting 0x22-0x23 Bias error level (digital data) 0x24-0x25 minimum Bias digital setting 0x26-0x27 maximum Bias digital setting 0x28-0x29 minimum Bias in milli-volts (analog) 0x2A-0x2B maximum Bias in milli-volts (analog) 0x2C-0x2C PSoC DAC channel to set Bias 0x2D-0x2D PSoC ADC channel to read Bias 0x2E-0x2F scale to convert ADC analog to mV	All Bias values are two-byte integers. Scale factor is two-bye unsigned integer that scales the ADC analog value (signed long) defined in bank #9 down to Bias in milli-volts.
4	0x30-0x3F	0x30-0x31 default ECRef data for driver 0 0x32-0x33 default ECRef data for driver 1 0x34-0x35 default ECRef data for driver 2 0x36-0x37 default ECRef data for driver 3 0x38-0x39 maximum ECRef setting 0x3A-0x3B ZIP code for setting ECRef data 0x3C-0x3D maximum Slopes setting 0x3E-0x3F ZIP code for setting Slopes	ECRef data and Slopes are two-byte unsigned integers; their minimum settings shall be 0.
5	0x40-0x4F	0x40-0x41 default Slope 0 setting for driver 0 0x42-0x43 default Slope 1 setting for driver 0 0x44-0x45 default Slope 2 setting for driver 0 0x46-0x47 default Slope 3 setting for driver 0 0x48-0x49 default Slope 4 setting for driver 0 0x4A-0x4B default Slope 5 setting for driver 0 0x4C-0x4D default Slope 6 setting for driver 0 0x4E-0x4F default Slope 7 setting for driver 0	Slope settings are two-byte unsigned integers.

Entry	Range	Data Format (16 bytes)	Notes
6	0x50-0x5F	0x50-0x51 default Slope 8 setting for driver 0 0x52-0x53 default Slope 9 setting for driver 0 0x54-0x55 default Slope 10 setting for driver 0 0x56-0x57 default Slope 11 setting for driver 0 0x58-0x59 default Slope 12 setting for driver 0 0x5A-0x5B default Slope 13 setting for driver 0 0x5C-0x5D default Slope 14 setting for driver 0 0x5E-0x5F default Slope 15 setting for driver 0	
7	0x60-0x6F	0x60-0x61 default Slope 0 setting for driver 1 0x62-0x63 default Slope 1 setting for driver 1 0x64-0x65 default Slope 2 setting for driver 1 0x66-0x67 default Slope 3 setting for driver 1 0x68-0x69 default Slope 4 setting for driver 1 0x6A-0x6B default Slope 5 setting for driver 1 0x6C-0x6D default Slope 6 setting for driver 1 0x6E-0x6F default Slope 7 setting for driver 1	
8	0x70-0x7F	0x70-0x71 default Slope 8 setting for driver 1 0x72-0x73 default Slope 9 setting for driver 1 0x74-0x75 default Slope 10 setting for driver 1 0x76-0x77 default Slope 11 setting for driver 1 0x78-0x79 default Slope 12 setting for driver 1 0x7A-0x7B default Slope 13 setting for driver 1 0x7C-0x7D default Slope 14 setting for driver 1 0x7E-0x7F default Slope 15 setting for driver 1	
9	0x80-0x8F	0x80-0x81 default Slope 0 setting for driver 2 0x82-0x83 default Slope 1 setting for driver 2 0x84-0x85 default Slope 2 setting for driver 2 0x86-0x87 default Slope 3 setting for driver 2 0x88-0x89 default Slope 4 setting for driver 2 0x8A-0x8B default Slope 5 setting for driver 2 0x8C-0x8D default Slope 6 setting for driver 2 0x8E-0x8F default Slope 7 setting for driver 2	
10	0x90-0x9F	0x90-0x91 default Slope 8 setting for driver 2 0x92-0x93 default Slope 9 setting for driver 2 0x94-0x95 default Slope 10 setting for driver 2 0x96-0x97 default Slope 11 setting for driver 2 0x98-0x99 default Slope 12 setting for driver 2 0x9A-0x9B default Slope 13 setting for driver 2 0x9C-0x9D default Slope 14 setting for driver 2 0x9E-0x9F default Slope 15 setting for driver 2	
11	0xA0-0xAF	0xA0-0xA1 default Slope 0 setting for driver 3 0xA2-0xA3 default Slope 1 setting for driver 3 0xA4-0xA5 default Slope 2 setting for driver 3 0xA6-0xA7 default Slope 3 setting for driver 3 0xA8-0xA9 default Slope 4 setting for driver 3 0xAA-0xAB default Slope 5 setting for driver 3 0xAC-0xAD default Slope 6 setting for driver 3 0xAE-0xAF default Slope 7 setting for driver 3	

Entry	Range	Data Format (16 bytes)	Notes
12	0xB0-0xBF	0xB0-0xB1 default Slope 8 setting for driver 3 0xB2-0xB3 default Slope 9 setting for driver 3 0xB4-0xB5 default Slope 10 setting for driver 3 0xB6-0xB7 default Slope 11 setting for driver 3 0xB8-0xB9 default Slope 12 setting for driver 3 0xBA-0xBB default Slope 13 setting for driver 3 0xBC-0xBD default Slope 14 setting for driver 3 0xBE-0xBF default Slope 15 setting for driver 3	
13	0xC0-0xCF	0xC0-0xC1 default Current DAC 0 data for driver 0 0xC2-0xC3 default Current DAC 0 data for driver 1 0xC4-0xC5 default Current DAC 0 data for driver 2 0xC6-0xC7 default Current DAC 0 data for driver 3 0xC8-0xC9 maximum Current DAC 0 setting 0xCA-0xCB ZIP code for setting Current DAC 0 0xCC-0xCD default Current DAC 1 data for driver 0 0xCE-0xCF default Current DAC 1 data for driver 1	Current DAC 0 and Current DAC 1 data (digital) are two-byte unsigned integers; their minimum settings shall be 0.
14	0xD0-0xDF	0xD0-0xD1 default Current DAC 1 data for driver 2 0xD2-0xD3 default Current DAC 1 data for driver 3 0xD4-0xD5 maximum Current DAC 1 setting 0xD6-0xD7 ZIP code for setting Current DAC 1 0xD8-0xDF TBD	
15	0xE0-0xEF	TBD	
16	0xF0-0xFF ADC channel 16	0xF0-0xF3 minimum analog value 0xF4-0xF7 maximum analog value 0xF8-0xF9 minimum digital data 0xFA-0xFB maximum digital data 0xFC-0xFF scale analog values to floats	Analog values are signed long. Digital data are signed short. Scale = [(analog long integer value) / (analog floating point value)].

Table 8-3 Serial EEPROM Bank #10

8.6 Serial EEPROM Bank #11

TBD. User has Read-Only access to this EEPROM bank.

Entry	Range	Data Format (16 bytes)	Notes
1	0x00-0x0F	GLV module PCB part number and rev. level: PWA-NNNNNN-NN-RR	N – digit (0-9) R – revision level
2	0x10-0x1F	GLV module PCB serial number: ANNNN-NNNNN-NNN	A – character (A-Z) N – digit (0-9)
3 - 16	0x20-0xFF	TBD	

8.7 Serial EEPROM Bank #12

TBD. User has Read-Only access to this EEPROM bank.

8.8 Serial EEPROM Bank #13

TBD. User has Read-Only access to this EEPROM bank.

8.9 Serial EEPROM Bank #14

TBD. User has Read-Only access to this EEPROM bank.

8.10 Serial EEPROM Bank #15

User has Read-Only access to this EEPROM bank.

Partition	Range	#Bytes	Data Content	Notes
1	0x00-0x3F	64	TBD.	
2	0x40-0x7F	64	TBD.	
3	0x80-0xBF	64	0x80-0x81 module-ON hours 0x82-0x83 hours @0 Celsius 0x84-0x85 hours @10 Celsius 0x86-0x87 hours @20 Celsius 0x88-0x89 hours @ 30 Celsius 0x8A-0x8B hours @40 Celsius 0x8C-0x8D hours @50 Celsius 0x8E-0x8F hours @60 Celsius 0x90-0x91 seconds in hour 0x92-0x93 PSoC D/A 0 setting 0x94-0x95 PSoC D/A 1 setting 0x96-0x97 PSoC D/A 2 setting 0x98-0x98 PSoC proc. status 0x99-0x99 GLV power level 0x9A-0x9A last error source 0x9B-0x9B last error number 0x9C-0x9C checksum of above 0x9D-0xBF TBD	This is the first of the two module-lifetime data sections (PING section). It is updated bi-hourly by PSoC firmware. The section with more "module-ON hours" is the most recent one. The one-byte checksum at 0x9C is just a direct sum of all the bytes from 0x80 to 0x9B (and keeping only the least-significant byte).
4	0xC0-0xFF	64	0xC0-0xC1 module-ON hours 0xC2-0xC3 hours @0 Celsius 0xC4-0xC5 hours @10 Celsius 0xC6-0xC7 hours @20 Celsius 0xC8-0xC9 hours @ 30 Celsius 0xCA-0xCB hours @40 Celsius 0xCC-0xCD hours @50 Celsius 0xCE-0xCF hours @60 Celsius 0xD0-0xD1 seconds in hour 0xD2-0xD3 PSoC D/A 0 setting 0xD4-0xD5 PSoC D/A 1 setting 0xD6-0xD7 PSoC D/A 2 setting 0xD8-0xD8 PSoC proc. status 0xD9-0xD9 GLV power level 0xDA-0xDA last error source 0xDB-0xDB last error number 0xDC-0xDC checksum of above 0xDD-0xFF TBD	This is the second of the two module-lifetime data sections (PONG section). It is updated bi-hourly by PSoC firmware. The section with more "module-ON hours" is the most recent one. The one-byte checksum at 0xDC is just a direct sum of all the bytes from 0xC0 to 0xDB (and keeping only the least-significant byte).

Table 8-4 Serial EEPROM Bank #15

9 Appendix F - GLV Optics and Alignment

The optical interface consists of the illumination and the imaging optics. The illumination optics couples the light source (laser or laser system) to the GLV, and the imaging optics images the modulated beam from the GLV to the target (e.g. photo resist-coated glass panel).

9.1 Illumination

The illumination optics convert a circular Gaussian beam produced by a laser into the line illumination profile required by the GLV with a footprint and an angular extent that are compatible with the GLV device operation. The illumination geometry is shown in **Figure 9-1** below.

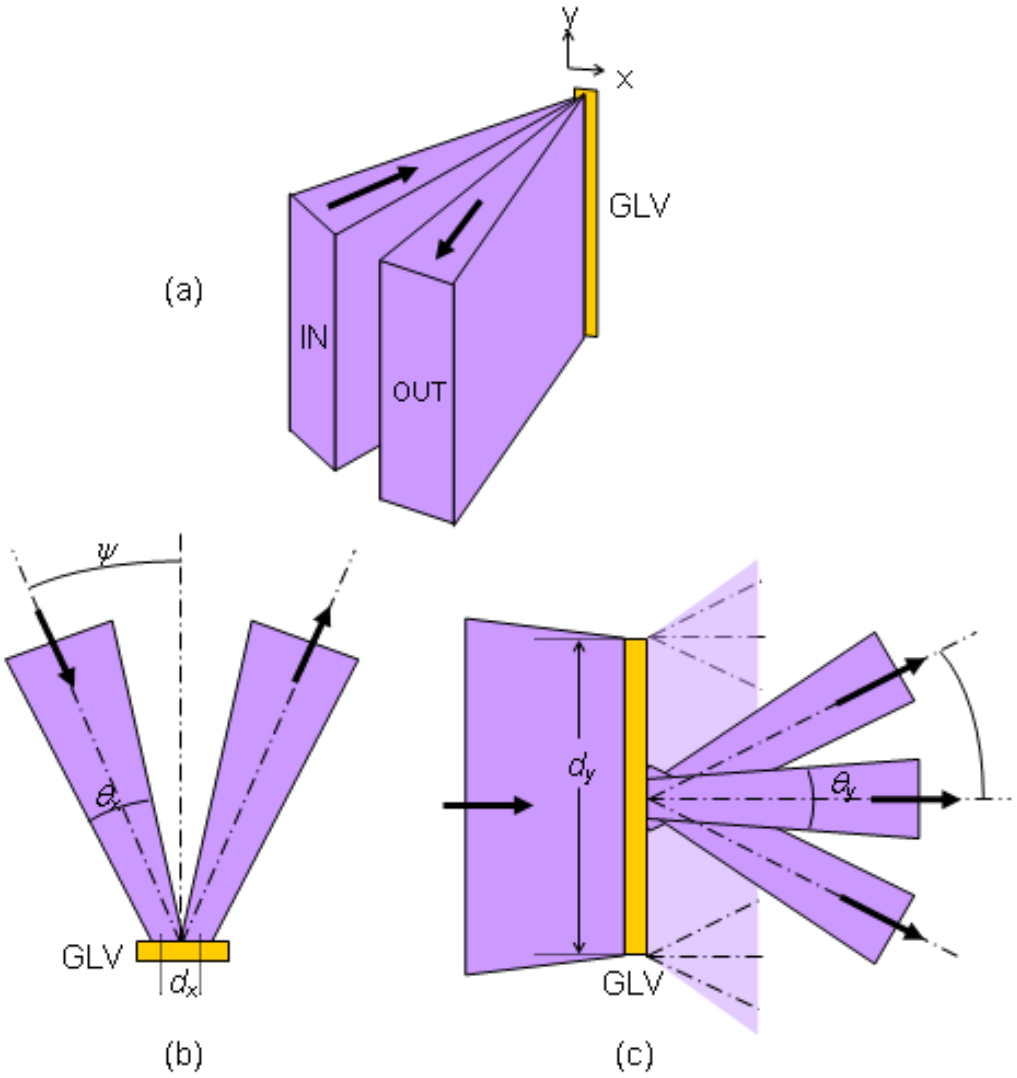

Figure 9-1 GLV illumination geometry: (a) perspective view, (b) "top" view, and (c) unfolded "side" view. The incoming & outgoing (reflected & diffracted) beams are indicated by the respective arrows.

Consider a GLV with *N* pixels, each of linear dimension *P*. A pixel corresponds to the minimum feature size in the image. Assume further that each pixel consists of *M* diffraction periods (or sub-pixel element) with period Λ. A period is formed by an active-bias ribbon pair. When a pixel consists of three subpixels, *M* = 3, *N* = 362 ("floor" of 1088/*M*), *P* = 25 μm and Λ = 8.5 μm (4.25um active + 4.25um bias). To facilitate further discussion, let's introduce a coordinate system with *x* along the GLV width, and *y* along its length as shown **Figure 9-1** (a). The desired illumination footprint at the GLV is a narrow rectangle, whose width is:

$$d_x \le P = M\Lambda \qquad (1)$$

And whose length is:

$$d_y \ge NP = NM\Lambda \quad . \qquad (2)$$

Beside the beam footprint, we also need to meet the angular constraints. In the x-direction, the illumination angular full-width θx must not exceed twice the incidence angle ψ as can be seed from **Figure 9-1**(b)

$$\theta_x \le 2\psi \quad , \qquad (3)$$

so that the incoming beam can be separated from the outgoing beam. Likewise, in the y-direction, the illumination angular full-width θy must be smaller or equal to the GLV diffraction angle θGLV as can be seen from **Figure 9-1**(c)

$$\theta_y \le \theta_{GLV} = \sin^{-1}\left(\frac{\lambda}{\Lambda}\right) \approx \frac{\lambda}{\Lambda} \quad , \qquad (4)$$

so that the 0th-order beam can be separated from the 1st-order beams. A more accurate expression that accounts for single pixel diffraction effect is

$$\theta_y \le \frac{M-2}{M}\frac{\lambda}{\Lambda} \quad , \qquad (5)$$

This is justified later in equation (9).

The beam that the illumination optics delivers to the GLV must satisfy (1), (2), (3) and (5). For the current GLV, with λ = 355 nm and ψ = 15°, dx $\le$ 25 μm, dy $\ge$ 27.7 mm, θx $\le$ 30° (in practice << 30°), and θy $\le$ 0.26°. The detail of the illumination optics depends of course on the specific light source and GLV parameters.

To efficiently illuminate the GLV, it is necessary that the source étendue be less than the GLV étendue. Étendue is defined as the spatial-angular product of the "light-bundle" that leaves the light source or incident on the light receiving object (GLV), and it is an optical invariant. A laser or multiple laser source generally have a sufficiently low étendue and can illuminate the GLV efficiently. A thermal source, on the other hand, are too wasteful to illuminate a GLV.

9.2 Imaging

The imaging optics selects the appropriate diffraction order(s), and maps the image at the target with the proper magnification.

The GLV device is essentially a phase modulator. If all diffraction orders are collected, the resulting image will have no contrast. The dominant power exchange occurs between the 0th and the ±1st orders. Since they carry complementary information, we employ either the 0^{th}-order or the 1^{st}-order operations in practice.

The typical GLV imaging optics for the 0^{th} and 1^{st} order operations are shown in Figure 9-2. There, FTL is the Fourier transform lens and IFTL the inverse Fourier transform lens.

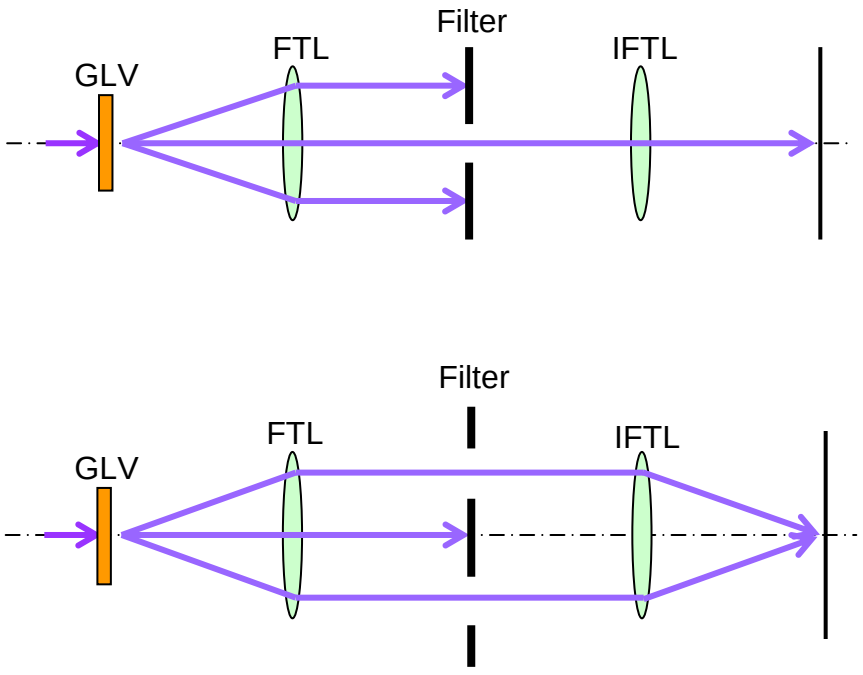

Figure 9-2 Typical imaging optics for (a) 0th order, and (b) 1st order operations.

The best location to discriminate the diffraction orders is the Fourier transform plane, where the spatial frequency spectrum of the image is formed. The Fourier transform plane coincides with the back focal plane of FTL. The diffraction order selection is done by placing a spatial filter (physically a set of apertures) that selects which order(s) to transmit, and which to block. For 0th-order operation, the spatial filter is simply an aperture (centered at the z-axis) that only transmits the modulated 0th-order.

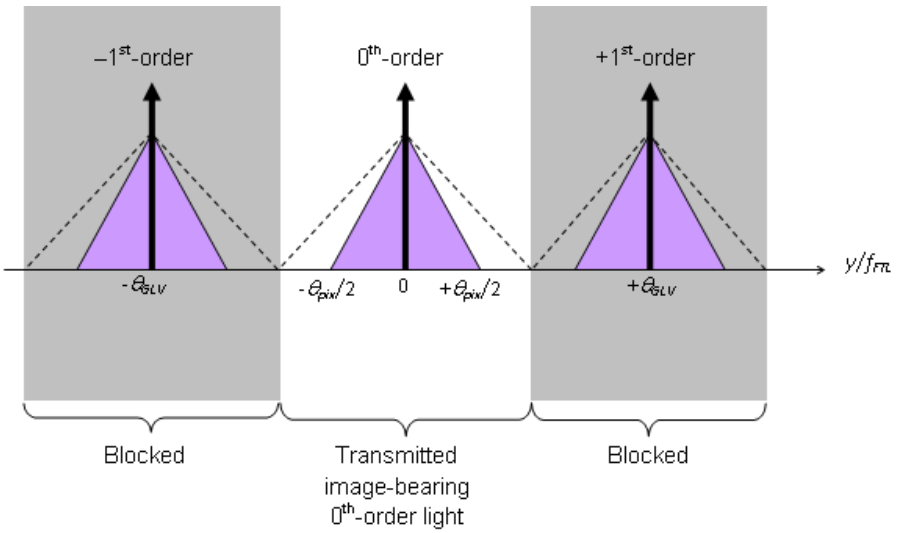

Figure 9-3 Spatial frequency spectrum at the Fourier transform plane.

Figure 9-3 shows the locations of the 0th and ±1st-order components along the (normalized) y-axis for the 1 pixel ON case (broadest modulation). Here,

$$\theta_{GLV} = \frac{\lambda}{\Lambda} \qquad (6)$$

$$\theta_{pix} = \frac{2\lambda}{M\Lambda} \qquad (7)$$

The opening of the 0th-order aperture extends to midway between the 0th and 1st orders. If the FTL focal length is fFTL , then the aperture diameter is

$$D_{aperture} = f_{FTL}\theta_{GLV} = f_{FTL}\frac{\lambda}{\Lambda} \quad . \qquad (8)$$

The room between two consecutive modulated orders allows illumination angular spread θy

$$\theta_y \le \theta_{GLV} - \theta_{pix} = \frac{M-2}{M}\frac{\lambda}{\Lambda} \quad , \qquad (9)$$

as mentioned earlier in equation (5). The filtered transform is converted to an intensity image by IFTL. The ratio of the IFTL focal length, fIFTL , and the FTL focal length determines the magnification m,

$$m = \frac{f_{IFTL}}{f_{FTL}} \quad . \qquad (10)$$

9.3 Optical Alignment Procedure

The detail of the alignment steps depend on the specific design of the GLV optical system. However, there are several general guiding principles that are common for any GLV optical system and these are explained below.

CAUTION: Make sure that all safety measures are implemented

We assume that:

- The GLV is mounted on a stage with all 6 degrees-of-freedom (DOF).

- The illumination beam meet all the conditions discussed in 3.1. Preferably, it also has the following DOFs: focus (z-translation) and axial rotation

- The FTL and IFTL, considered as a single unit, is well aligned. Preferably, it also has the following DOFs: overall image focus on target and z translation DOF of the spatial filter

STEP 1 - Preparation

a) Insert a reference HeNe laser beam to establish the optical z-axis. This should be done with the GLV and all mirrors in their nominal positions, but without any refractive component. We may need to activate the GLV to generate the 0th-order beam. The best location to insert the HeNe beam is as upstream as possible toward the laser. The HeNe beam insertion must be consistently repeatable and can be done, for example, by using a "flipper" mirror. A beam splitter is not recommended since it will deviate the beam when removed.

b) Adjust the GLV and all mirrors so that the HeNe beam propagation coincides with the designed optical z-axis. In particular, adjust the incident angle at the GLV to meet the specified value, e.g. $\psi = 15^\circ$.

c) Place iris diaphragms at several locations to mark where the beam axis should go. Ideally, there is a pair of iris diaphragms for each optic component, one placed before and the other after the component. Although the insertion of an optic component may diverge or converge the beam, the beam center should be aligned to coincide with the centers of all iris diaphragms. (in practice, the divergence / convergence imparted by the component is small for a ~1 mm HeNe beam)

STEP 2 – Optics insertion

a) Place the imaging optics (FTL/IFTL pair) without the spatial filter in its nominal location. Adjust so that the emerged HeNe beam is centered at the z-axis.

b) Switch to the illumination beam. Use attenuated power. Adjust the beam orientation axially so that the beam (local) y-axis is parallel with the GLV y-axis. Adjust so that the beam, everywhere from the laser to the target, is centered at the z-axis by inspection on the iris diaphragms. At this stage, keep the illumination beam somewhat out-of-focus at the GLV. This "flood" illumination makes the next step easier.

STEP 3 – Imaging optics alignment

a) This step establishes the imaging of the GLV to the target. Align the GLV, imaging optics, and the target (if permitted), so that the GLV and the target are conjugate, i.e. they are in object-image relationship. Adjust GLV in x and y so the middle pixel is on the z-axis and is mapped to the image center at the target plane. Rotate the GLV axially about the z-axis so that the two fiducial crosses at both GLV ends are on the y-axis of the target. Repeat 2 b) if necessary.

b) Place the 0th-order spatial filter at the transform plane. Align the spatial filter so that the GLV 0th-order diffraction go through the center of the filter aperture.

STEP 4 – Illumination optics alignment

a) Translate the illumination optics along the z-axis until the beam waist coincides with the GLV. It is easy to check this condition by making sure that the line image at the target plane is narrowest. At this condition, the x-width of the orders at the transform plane will also reach a maximum.

STEP 5 – Final overall alignment

a) Although use in 0th-order operation, it will be convenient during the alignment to switch back-and-forth between both orders.

b) The 1st-order operation, with spatial filter in Fig. 3.1.1 (b), provides a fast and accurate way to determine conjugacy. With three (or more) one pixels diffracting (one at the center, and one toward each GLV ends), the proper conjugate is reached if the images of the two orders totally overlap at the image plane. Make sure that the overlap is met simultaneously for the three (or more) pixels. Adjustment can be done by fine rotation of the GLV about the x-axis.

c) Set all pixels ON in the 0th-order. The modulation will be maximum at the ribbon center (avoid roll-over) and diminishes away from the center. Translate the GLV back-and-forth in the x-direction and make sure that the modulation decreases uniformly across the entire length of the GLV. Otherwise, correct the relative axial rotation between the GLV and the illumination.

d) Check the y-thickness of the orders at the transform plane. For all pixels ON, the orders must have minimum thickness. This can be tested by comparing the thickness at different z location in the transform plane neighborhood. Non minimum thickness indicates that the "side-view" illumination beam is not properly collimated and should be corrected. In marginal situation, improper collimation will cause order overlap at the transform plane, resulting in image degradation and/or reduced contrast.

e) Check the orders at the transform plane. The 0th-order should be centered at the z-axis. Fine-tuning can be accomplished by rotation of the GLV about the x-axis. If needed, fine adjust the spatial filter placement so that it is filtering the diffraction orders symmetrically.

f) Iterate above steps if necessary.

10 Product Specifications

Refer to SLM Document “High-Speed Buddy GLV Module Specification”, SLM document number SPC-006680-XX.

11 References

Please refer to the following documents for more detailed information.

- *Silicon Light Machine – White Papers*
  - http://www.siliconlight.com/en/technology/media-white-papers.html
  - *Silicon Light Machines – Grating Light Valve Technology Brief*
- *“The I²C-Bus Specification”; Phillips Semiconductor*
- *Data Sheet for PSoC titled: “CY8C27143, CY8C27243, CY8C27443, CY8C27543, CY8C27643 PSoC Programmable System on a Chip Document Number: 38-12012”*
- *Data Sheet for LVDS Serializer titled: “DS90CR485: 133MHz 48-Bit Channel Link Serializer (6.384 Gbps)”; National Semiconductor / TI*
- *Data Sheet for LVDS Deserializer titled: “DS90CR486: 133MHz 48-Bit Channel Link Deserializer (6.384 Gbps)”; National Semiconductor / TI*
- *Data Sheet for LVDS Receiver titled: “FIN1048 3.3V LVDS 4-Bit Flow-Through High Speed Differential Receiver”; Fairchild Semiconductor*
- *Data Sheet for I2C Buffer titled: “P82B96: Dual bidirectional bus buffer”; NXP*

12 Revision History

Rev	Authors	Description of change	Date
1-A	Greg Myatt, Alex Payne, Anatoly Volchegursky, Yansun Xu	Original release	03/31/16
1-B	Greg Myatt, Yansun Xu	Section 3.1.3: "Cypress Microsystem's" changed to "Cypress Semiconductor's". Table 3-3: Serializer Clock Frequency was increased from 104 to 108 MHz and clock frequencies were increased accordingly. Table 3-10 & Table 3-11: Serializer Clock Frequency was increased from 104 to 108 MHz; "Time for Column" was broken down into the following components: Last Data Word to Column Strobe, Column Pulse Width & Controller Allocation. Number of clocks was reduced by 8 since propagation delay through the SLM212 driver had been accounted for twice in the calculator. Notes were added to text to support the tables. Table 3-12 & Table 3-13: "Packet to Column Timing Specification" Serializer Clock Frequency was increased from 104 to 108 MHz; tC2F parameter symbol was corrected to tC2Z for Simple Timing Mode; Expected column rate increase from simple timing mode to advanced timing mode was decreased from 7% to 6%. Section 3.2.1: Serializer Clock Frequency was increased from 104 to 108 MHz. Section 3.2.2: Serializer Clock Frequency was increased from 104 to 108 MHz and maximum column rate for simple timing mode was increased to 350 kHz. "Time for Column" was broken down into the following components: Last Data Word to Column Strobe, Column Pulse Width & Controller Allocation. Table 3-26: Bit 4 changed from VREF1.3 to "Not Used". Table 3-30: Added status code -3, -6 & modified code -4. Section 3.3.1: In Step 9a, the command byte should be 0xE5 (not 0xE3) for "Reset FPGA" command. Table 5-17: tC2D was changed from -2tCPD to 0 nsec. Section 10: SLM document number corrected to SPC-006680-XX.	06/16/16