This user guide describes the I^{2}C Interface channel setup, protocol details, and supported commands to configure and control the OLED Driver.

1 Channel Configuration

Slave addresses:	0x74, 0x75	
Bus speed	Standard rate:	up to 100 kbps
	Fast Mode:	up to 400 kbps
Data size	1 or 2 bytes (see specific register definitions)	

2 Register Transactions

In all cases, the OLED Driver is ready to process an I^{2}C command when the I^{2}C bus is idle with SCL high (ref: Appendix A).

2.1 Single byte

Single-byte parameters must be written / read in standard 1-byte protocol by performing a 1-data-byte write / read from the specified register addresses (lower address) in **bold** in Command Table.

2.2 Double byte

Double-byte parameters must be written / read in standard 2-byte protocol by performing a 2-data-byte write / read from the lower of the two specified register addresses in **bold** in Command Table.

2.2.1 Transaction Data Formatting

2.2.2 Unsigned Integer

Data encoding:	hexadecimal	
Data size:	2 bytes (16 bits)	
	Lower address byte:	least significant byte
	Higher address byte:	most significant byte

2.2.3 Signed Integer

Data encoding:	2's complement	
Data size:	2 bytes (16 bits)	
	Lower address byte:	least significant byte
	Higher address byte:	most significant byte

2.2.4 Unsigned Floating Point (f-p)

Data encoding:	hexadecimal	
Data size:	2 bytes (16 bits)	
	Lower address byte:	fraction * 100
	Higher address byte:	integer

2.2.5 Signed Floating Point (f-p)

Data encoding:	2's complement	
Data size:	2 bytes (16 bits)	
	Lower address byte:	fraction * 100
	Higher address byte:	integer

3 Command Table

<table>
<tr><th colspan="5">Register Info</th><th colspan="6">Argument (all numerical values in decimal)</th><th colspan="3">Description</th></tr>
<tr><th>Label</th><th>Addr</th><th>r/w</th><th>Index</th><th>Size (byte)</th><th>min</th><th>Factory Default [1]</th><th>max</th><th>Type</th><th>unit</th><th>Auto-Save</th><th>Class</th><th colspan="2">Function</th></tr>
<tr><td>IVQ_INFO</td><td>0x00</td><td>r</td><td>n/a</td><td>1</td><td>-</td><td>-</td><td>-</td><td>binary</td><td>n/a</td><td>n/a</td><td>Input Control</td><td colspan="2">[7] LOCK 0 = Unlocked, 1 = Locked
[6] *reserved* -
[5:4] SDV_STAT [00] INPUT = None detected
[01] NTSC Detected
[10] PAL Detected
[11] *resereved*
[3:2] *reserved* -
[1:0] INPUT [00] INPUT = None
[01] INPUT = SDV Analog
[10] INPUT = SDV Digital / BT.656
[11] INPUT = VESA</td></tr>
<tr><td>IVQ_HA</td><td>0x01
0x02</td><td>r</td><td>n/a</td><td>2</td><td>-</td><td>-</td><td>-</td><td>unsigned int</td><td>Pixel</td><td>n/a</td><td>Input Control</td><td colspan="2">[7:0] Least Significant byte
[7:0] Most significant byte</td></tr>
<tr><td>IVQ_VA</td><td>0x03
0x04</td><td>r</td><td>n/a</td><td>2</td><td>-</td><td>-</td><td>-</td><td>unsigned int</td><td>line</td><td>n/a</td><td>Input Control</td><td colspan="2">[7:0] Least Significant byte
[7:0] Most significant byte</td></tr>
<tr><td>IVQ_FP</td><td>0x05
0x06</td><td>r</td><td>n/a</td><td>2</td><td>-</td><td>-</td><td>-</td><td>unsigned f-p</td><td>MHz</td><td>n/a</td><td>Input Control</td><td colspan="2">[7:0] Fraction * 100
[7:0] Integer</td></tr>
<tr><td>SDVA</td><td>0x07</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>-00, -01: **0**
-10, -11: **1**</td><td>1</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Input Control</td><td colspan="2">Set video input format for analog SDV input
0 = Y, CVBS, 1 = Y/C</td></tr>
<tr><td>SDVD</td><td>0x08</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>-00, -10: **0**
-01, -11: **1**</td><td>1</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Input Control</td><td colspan="2">Set video input format for digital SDV input
0 = NTSC / YCbCr 4:2:2, 1 = PAL / YCbCr 4:2:2</td></tr>
<tr><td>BRT</td><td>0x09</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>**8**</td><td>BRTMOD</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td colspan="2">Set brightness (luminance). Range = 0 ~ BRTMOD.</td></tr>
<tr><td>BRTMOD</td><td>0x0A</td><td>r/w</td><td>n/a</td><td>1</td><td>4</td><td>**16**</td><td>32</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td colspan="2">Set number of BRT adjustment steps (modulus)</td></tr>
<tr><td>BRTINC</td><td>0x0B</td><td>w</td><td>n/a</td><td>1</td><td>0</td><td>n/a</td><td>1</td><td>n/a</td><td>n/a</td><td>n/a</td><td>Image Control</td><td colspan="2">Increment BRT
0 = No operation, 1 = Increment BRT by 1</td></tr>
<tr><td>BRTDEC</td><td>0x0C</td><td>w</td><td>n/a</td><td>1</td><td>0</td><td>n/a</td><td>1</td><td>n/a</td><td>n/a</td><td>n/a</td><td>Image Control</td><td colspan="2">Decrement BRT
0 = No operation, 1 = Decrement BRT by 1</td></tr>
<tr><td>MAXLUM</td><td>0x0D</td><td>r/w</td><td>n/a</td><td>1</td><td>10</td><td>**100**</td><td>100</td><td>unsigned int</td><td>%</td><td>Yes</td><td>Image Control</td><td colspan="2">Set maximum display luminance, **100%** = display device upper limit</td></tr>
<tr><td>GAMMA</td><td>0x0E
0x0F</td><td>r/w</td><td>n/a</td><td>2</td><td>1.0</td><td>**1.8**</td><td>2.2</td><td>unsigned f-p</td><td>exp</td><td>Yes</td><td>Image Control</td><td colspan="2">[7:0] Fraction * 100
[7:0] Integer</td></tr>
<tr><td>COLOR</td><td>0x10</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>**4**</td><td>4</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td colspan="2">Set color mode
Color OLED (BIT1011B1):
<table>
<tr><th></th><th colspan="2">SDV</th><th>VESA</th><th></th></tr>
<tr><th></th><th>analog</th><th>digital</th><th>digital</th><th>Notes</th></tr>
<tr><td>0</td><td>blk/w</td><td>n/a</td><td>blk/w</td><td>digital SDV displays color</td></tr>
<tr><td>1</td><td>blk/r</td><td>blk/r</td><td>blk/r</td><td rowspan="3">SDV Digital: true monochrome only for monochrome input, else “off” colors are suppressed</td></tr>
<tr><td>2</td><td>blk/g</td><td>blk/g</td><td>blk/g</td></tr>
<tr><td>3</td><td>blk/b</td><td>blk/b</td><td>blk/b</td></tr>
<tr><td>**4**</td><td>**color**</td><td>**color**</td><td>**n/a**</td><td>**VESA displays blk/wht**</td></tr>
</table>
Monochrome OLED (BIT1011B2):
<table>
<tr><th></th><th colspan="2">SDV</th><th>VESA</th><th></th></tr>
<tr><th></th><th>Analog</th><th>Digital</th><th>Digital</th><th>Notes</th></tr>
<tr><td>0</td><td rowspan="5" colspan="2">Monochrome in color of OLED</td><td>full luminance</td><td></td></tr>
<tr><td>1</td><td rowspan="3">reduced luminance</td><td rowspan="3"></td></tr>
<tr><td>2</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td>**full luminance**</td><td></td></tr>
</table></td></tr>
<tr><td>HPOS</td><td>0x11
0x12</td><td>r/w</td><td>n/a</td><td>2</td><td>[3] -92
[4] -12</td><td>**0**</td><td>[3] +92
[4] +12</td><td>signed int</td><td>column</td><td>Yes</td><td>Image Control</td><td colspan="2">Set horizontal image position offset
0 = default center neutral</td></tr>
<tr><td>VPOS</td><td>0x13
0x14</td><td>r/w</td><td>n/a</td><td>2</td><td>[3] -72
[4] -12</td><td>**0**</td><td>[3] +72
[4] +12</td><td>signed int</td><td>row</td><td>Yes</td><td>Image Control</td><td colspan="2">Set vertical image position offset
0 = default center neutral</td></tr>
<tr><td>HSCAN</td><td>0x15</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>**0**</td><td>1</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td colspan="2">Set horizontal image scan direction
0 = normal L → R, 1 = reversed R → L</td></tr>
<tr><td>VSCAN</td><td>0x16</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>**0**</td><td>1</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td colspan="2">Set vertical image scan direction
0 = normal T → B, 1 = reversed B → T</td></tr>
<tr><td>DPDC</td><td>0x17</td><td>r/w</td><td>n/a</td><td>1</td><td>1</td><td>**100**</td><td>100</td><td>unsigned int</td><td>%</td><td>Yes</td><td>Image Control</td><td colspan="2">Set frame duty-cycle (**100% = no row blanking**)</td></tr>
<tr><td>OPATT</td><td>0x18</td><td>r/w</td><td>n/a</td><td>1</td><td>0</td><td>**0**</td><td>8</td><td>unsigned int</td><td>n/a</td><td>No</td><td>Image Control</td><td colspan="2">Set OLED Driver Test Pattern
0 = No Pattern
1 = White Field
2= Color Bar
3 = Gray Scale
4 = Checker Board
5 = Vertical Lines
6 = Horizontal Lines
7 = Grid Pattern
8 = Black Field</td></tr>
<tr><td>WBCR</td><td>0x19
0x1A</td><td>r/w</td><td>n/a</td><td>2</td><td>0</td><td>**256**</td><td>256</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td>[7:0] Least Significant byte
[7:0] Most significant byte</td><td>Set Red channel pixel gain:
0 = 0%, **256 = 100%**</td></tr>
<tr><td>WBCG</td><td>0x1B
0x1C</td><td>r/w</td><td>n/a</td><td>2</td><td>0</td><td>**256**</td><td>256</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td>[7:0] Least Significant byte
[7:0] Most significant byte</td><td>Set Green channel pixel gain:
0 = 0%, **256 = 100%**</td></tr>
<tr><td>WBCB</td><td>0x1D
0x1E</td><td>r/w</td><td>n/a</td><td>2</td><td>0</td><td>**256**</td><td>256</td><td>unsigned int</td><td>n/a</td><td>Yes</td><td>Image Control</td><td>[7:0] Least Significant byte
[7:0] Most significant byte</td><td>Set Blue channel pixel gain:
0 = 0%, **256 = 100%**</td></tr>
<tr><td>VER</td><td>0x1F
0x20</td><td>r</td><td>n/a</td><td>2</td><td>-</td><td>**-**</td><td>-</td><td>unsigned f-p</td><td>n/a</td><td>n/a</td><td>System</td><td>[7:0] Fraction * 100
[7:0] Integer</td><td>Read SW Version Number</td></tr>
</table>

Register Info					Argument (all numerical values in decimal)						Description	
Label	Addr	r/w	Index	Size (byte)	min	Factory Default [1]	max	Type	unit	Auto-Save	Class	Function
POWER	**0x21**	r/w	n/a	1	0	**1**	1	unsigned int	n/a	Yes	System	Set power mode 0 = powered-down (low-power), **1 = operational**
LEDEN	**0x22**	r/w	n/a	1	0	**1**	1	unsigned int	n/a	Yes	System	Set LED enable: 0 = Status LEDs disabled, **1 = Status LEDs enabled**
OMONOCHR	**0x23**	r/w	n/a	1	0	BIT1011B1: **0** BIT1011B2: **1**	1	unsigned int	n/a	Yes	System	Set OLED type: 0 = Color XL, 1 = Monochrome Green XLT
OTEMP	**0x24** 0x25	r	n/a	2	n/a	n/a	n/a	signed f-p	°C	n/a	System	[7:0] Fraction * 100 [7:0] Integer Query OLED Temperature
OLEDSN_0	**0x26**	r	n/a	1	n/a	n/a	n/a	ASCII	n/a	n/a	System	Returns OLED Serial Number Char 1 (Leftmost)
OLEDSN_1	**0x27**	r	n/a	1	n/a	n/a	n/a	ASCII	n/a	n/a	System	Returns OLED Serial Number Char 2
OLEDSN_2	**0x28**	r	n/a	1	n/a	n/a	n/a	ASCII	n/a	n/a	System	Returns OLED Serial Number Char 3
OLEDSN_3	**0x29**	r	n/a	1	n/a	n/a	n/a	ASCII	n/a	n/a	System	Returns OLED Serial Number Char 4
OLEDSN_4	**0x2A**	r	n/a	1	n/a	n/a	n/a	ASCII	n/a	n/a	System	Returns OLED Serial Number Char 5 (Righftmost)

[1] Factory Default values overwritten upon change by auto-save function.

4 Detailed Register Descriptions

4.1 Input Control Registers

4.1.1 IVQ_INFO

Register address: **0x00**
Access type: read-only
Data type: single byte
Function: Reads input video status information.

[7] **SDV_LOCK** Indicates lock status of analog SDV decoder
0 = unlocked
1 = locked

[6] *reserved*

[5:4] **SDV_STAT** Indicates SDV input detection status
[00] None Detected
[01] NTSC Detected
[10] PAL Detected
[11] *reserved*

[3:2] *reserved*

[1:0] **INPUT** Indicates INPUT configuration
[00] INPUT = None
[01] INPUT = SDV - Analog
[10] INPUT = SDV - Digital (BT.656)
[11] INPUT = VESA

4.1.2 IVQ_HA

Register addresses: **0x01**, **0x02**
Access type: read-only
Data type: double byte unsigned integer
Function: Reads input video Horizontal Active period (in pixels).

4.1.3 IVQ_VA

Register addresses: **0x03**, **0x04**
Access type: read-only
Data type: double byte unsigned integer
Function: Reads input video Vertical Active period (in lines).

4.1.4 IVQ_FP

Register addresses: **0x05**, **0x06**
Access type: read-only
Data type: double byte floating point
Function: Reads input video Pixel Clock Frequency (in MHz).

4.1.5 SDVA

Register address:	**0x07**
Access type:	write / read
Data type:	single byte
Function:	Sets video format for analog SDV input.

SDVA = 0	Monochrome: Y Color: CVBS
SDVA = 1	Color: Y/C

4.1.6 SDVD

Register address:	**0x08**
Access type:	write / read
Data type:	single byte
Function:	Sets video format for digital SDV input (BT.656).

SDVD = 0	NTSC
SDVD = 1	PAL

4.2 Image Control Registers

4.2.1 BRT

Register address:	**0x09**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets display luminance.

Range: 0 ≤ **BRT** ≤ **BRTMOD**.

4.2.2 BRTMOD

Register address:	**0x0A**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets the number of **BRT** steps over luminance range (modulus).

Example:

BRTMOD	= 16
BRT steps	= 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, **16**

4.2.3 BRTINC

Register address:	**0x0B**
Access type:	write-only
Data type:	single byte unsigned integer
Function:	Increments **BRT** value by 1. If **BRT** = **BRTMOD**, no operation.

BRTINC = 0	no action
BRTINC = 1	increment BRT

4.2.4 BRTDEC

Register address:	**0x0C**
Access type:	write-only
Data type:	single byte unsigned integer
Function:	Decrements **BRT** value by 1. If **BRT** = **0**, no operation.

BRTDEC = 0	no action
BRTDEC = 1	decrement BRT

4.2.5 MAXLUM

Register address:	**0x0D**
Access type:	write / read
Data type:	single byte unsigned integer

Function:	Sets luminance value for **BRT** = **BRTMOD** as a percentage of maximum OLED luminance.

4.2.6 GAMMA

Register addresses:	**0x0E**, **0x0F**
Access type:	write / read
Data type:	double byte unsigned floating-point
Function:	Sets display gamma value.

Example:

GAMMA = 1.8 = 0x50, 0x01

4.2.7 COLOR

Register address:	**0x10**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets display color mode.

COLOR = 0 Monochrome:

BIT1011B1 w/ Color OLED:	blk / wht
BIT1011B2 w/ Monochrome OLED:	monochrome in color of OLED, full luminance

COLOR = 1 Monochrome

BIT1011B1 w/ Color OLED:	blk / red
BIT1011B2 w/ Monochrome OLED:	monochrome in color of OLED, reduced luminance

COLOR = 2 Monochrome

BIT1011B1 w/ Color OLED:	blk / grn
BIT1011B2 w/ Monochrome OLED:	monochrome in color of OLED, reduced luminance

COLOR = 3 Monochrome

BIT1011B1 w/ Color OLED:	blk / blu
BIT1011B2 w/ Monochrome OLED:	monochrome in color of OLED, reduced luminance

COLOR = 4 Color

BIT1011B1 w/ Color OLED:	full color
BIT1011B2 w/ Monochrome OLED:	monochrome in color of OLED, full luminance

4.2.8 HPOS

Register addresses:	**0x11**, **0x12**
Access type:	write / read
Data type:	double byte signed integer
Function:	Sets horizontal image position.

Negative values move image to the left, positive values move image to the right.
Unit = 1 column (pixel)

4.2.9 VPOS

Register addresses:	**0x13**, **0x14**
Access type:	write / read
Data type:	double byte signed integer
Function:	Sets vertical image position.

Negative values move image up, positive values move image down.
Unit = 1 row (line)

4.2.10 HSCAN

Register address:	**0x15**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets horizontal scan direction.

HSCAN = 0 normal horizontal scan direction (L → R)

HSCAN = 1	reverse horizontal scan direction (L ← R)

Normal scan orientation specified in product datasheet.

4.2.11 VSCAN

Register address:	**0x16**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets vertical scan direction.

VSCAN = 0	normal vertical scan direction (T → B)
VSCAN = 1	reverse vertical scan direction (T ← B)

Normal scan direction specified in product datasheet.

4.2.12 DPDC

Register address:	**0x17**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets Display PWM Duty Cycle.

4.2.13 OPATT

Register address:	**0x18**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets bult-in test pattern.

OPATT = 0	No pattern selected, input video displayed
OPATT = 1	White Field
OPATT = 2	Color bar
OPATT = 3	Gray scale
OPATT = 4	Checker board
OPATT = 5	Vertical lines
OPATT = 6	Horizontal lines
OPATT = 7	Grid pattern
OPATT = 8	Black Field

4.2.14 WBCx

Register addresses:	Red:	**0x19**, **0x1A**
	Green:	**0x1B**, **0x1C**
	Blue:	**0x1D**, **0x1E**
Access type:	write / read	
Data type:	double byte unsigned integer	
Function:	With BIT1011B1 + Color OLED, sets R/G/B channel gain values of White Balance Control.	

x = R, G, B

The White Balance Control (**WBC**) function provides a method for adjusting the R-G-B mix of video data applied to the OLED to establish a desired white chromaticity value by attenuating excessively strong color component(s). The applied R/G/B color pixel values are as follows:

$\text{RED}_{\text{applied}}$ = RED * (**WBCR**/256)
$\text{GRN}_{\text{applied}}$ = GRN * (**WBCG**/256)
$\text{BLU}_{\text{applied}}$ = BLU * (**WBCB**/256)

With BIT1011B2 + Monochrome OLED, redudes luminance.

4.3 System Registers

4.3.1 VERSION

Register addresses:	**0x1F**, **0x20**
Access type:	read-only
Data type:	double byte unsigned floating-point

Function:	Reads software version number.

Example (acutal SW version may vary):

Lower address byte =	0x04
Upper address byte =	0x01
SW version =	**1.04**

4.3.2 POWER

Register address:	**0x21**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets power mode.

POWER =0:	Power down command / OLED Driver is in powered-down state
1:	Power up command / OLED Driver is in operational state

4.3.3 LEDEN

Register address:	**0x22**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets **ACT**, **OOL** indicator LED mode.

LEDEN = 0	LEDs disabled
1	LEDs enabled

4.3.4 OMONOCHR

Register address:	**0x23**
Access type:	write / read
Data type:	single byte unsigned integer
Function:	Sets OLED compatibility type.

OMONOCHR = 0	Color
1	Monochrome Green

4.3.5 OTEMP

Register address:	**0x24, 0x25**
Access type:	read-only
Data type:	double byte signed integer
Function:	Reads OLED temperature (in Celsius).

Example (acutal OLED temperature will vary):

Lower address byte =	0x4C
Upper address byte =	0x33
OLED Temperature =	**51.76** °C

4.3.6 OLEDSN_0 ~ _4

Register addresses:	**0x26, 0x27, 0x28, 0x29, 0x2A**
Access type:	read-only
Data type:	single byte ASCII
Function:	Reads 5 character alphanumeric OLED serial number.

Example (acutal OLED SN will vary):

Lowest address byte (leftmost character)	addr 0x26:	0x5A =	**Z**
Next address byte	addr 0x27:	0x31 =	**1**
Next address byte	addr 0x28:	0x30 =	**0**
Next address byte	addr 0x29:	0x56 =	**V**
Highest address byte (rightmost character)	addr 0x2A:	0x50 =	**P**

OLED Serial Nuber = **Z10VP**

Contact

Bild Innovative Technology LLC
2656 W Montrose Ave Suite 203
Chicago, IL 60618
(773) 697-9345
(760) 450-6633
pbeerdeke@bildinnotech.com

APENDIX A
I2C Bus Operation

A1 I²C Interface

A1.1 General I²C Operation

The I²C bus is a standard bidirectional interface that uses a controller, known as the master, to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master. Each device on the I²C bus has a specific device address to differentiate between other devices that are on the same I²C bus. Many slave devices will require configuration upon startup to set the behavior of the device. This is typically done when the master accesses the slave's internal register maps, which have unique register addresses. A device can have one or multiple registers where data is stored, written, or read.

The physical I²C interface consists of the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines must be connected to V_{CC} through a pull-up resistor. The size of the pull-up resistor is determined by the amount of capacitance on the I²C . Data transfer may be initiated only when the bus is idle. A bus is considered idle if both SDA and SCL lines are high after a STOP condition.

The general procedure for a master to access a slave device is the following:

1. Suppose a master wants to send data to a slave:
   - Master-transmitter sends a START condition and addresses the slave-receiver
   - Master-transmitter sends data to slave-receiver
   - Master-transmitter terminates the transfer with a STOP condition
2. If a master wants to receive/read data from a slave:
   - Master-receiver sends a START condition and addresses the slave-transmitter
   - Master-receiver sends the requested register to read to slave-transmitter
   - Master-receiver receives data from the slave-transmitter
   - Master-receiver terminates the transfer with a STOP condition

A1.1.1 START and STOP Conditions

I²C communication with this device is initiated by the master sending a START condition and terminated by the master sending a STOP condition. A high-to-low transition on the SDA line while the SCL is high defines a START condition. A low-to-high transition on the SDA line while the SCL is high defines a STOP condition.

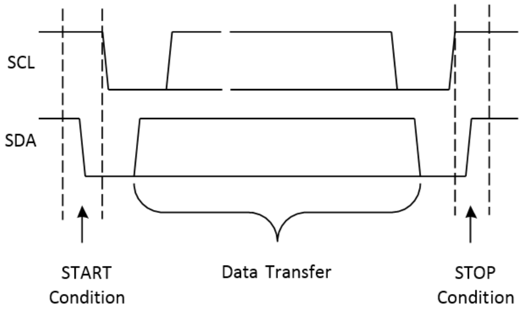

Figure A1. Example of START and STOP Condition

A1.1.2 Repeated START Condition

A repeated START condition is similar to a START condition and is used in place of a back-to-back STOP then START condition. It looks identical to a START condition, but differs from a START condition because it happens before a STOP condition (when the bus is not idle). This is useful for when the master wishes to start a new communication, but does not wish to let the bus go idle with the STOP condition, which has the chance of the master losing control of the bus to another master (in multi-master environments).

A1.2 Data Validity and Byte Format

One data bit is transferred during each clock pulse of the SCL. One byte is comprised of eight bits on the SDA line. A byte may either be a device address, register address, or data written to or read from a slave. Data is transferred Most Significant Bit (MSB) first. Any number of data bytes can be transferred from the master to slave between the START and STOP conditions. Data on the SDA line must remain stable during the high phase of the clock period, as changes in the data line when the SCL is high are interpreted as control commands (START or STOP).

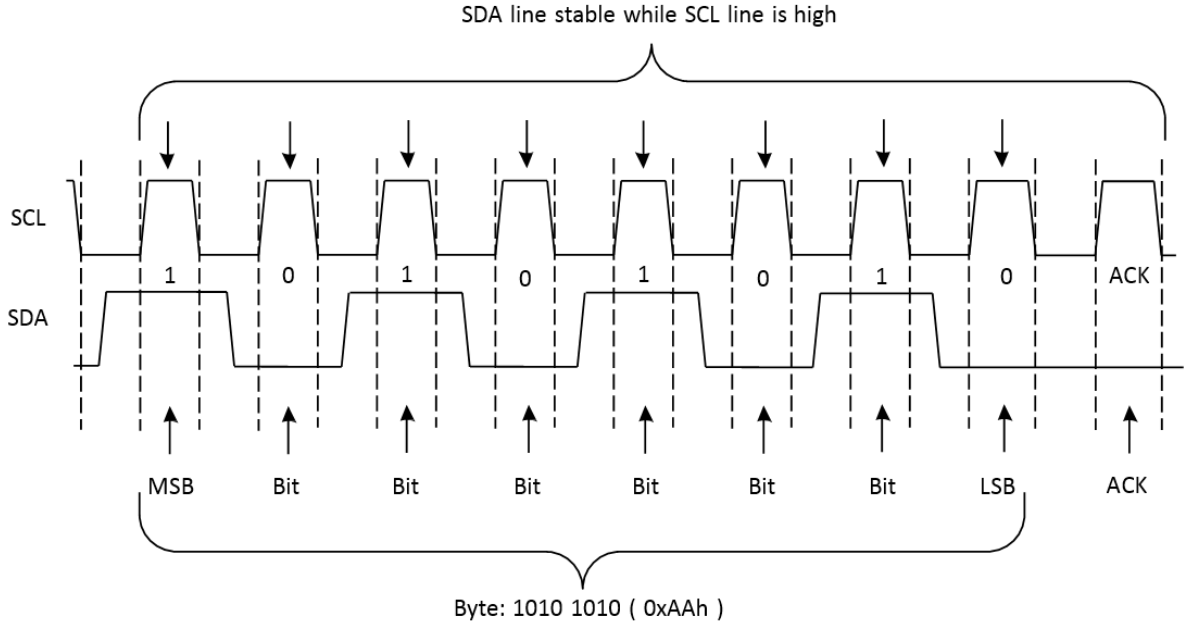
SDA line stable while SCL line is high
SCL
SDA
1
0
1
0
1
0
1
0
ACK
MSB
Bit
Bit
Bit
Bit
Bit
Bit
LSB
ACK
Byte: 1010 1010 (0xAAh)

Figure A2. Example of Single Byte Data Transfer

A1.3 Acknowledge (ACK) and Not Acknowledge (NACK)

Each byte of data (including the address byte) is followed by one ACK bit from the receiver. The ACK bit allows the receiver to communicate to the transmitter that the byte was successfully received and another byte may be sent.

Before the receiver can send an ACK, the transmitter must release the SDA line. To send an ACK bit, the receiver shall pull down the SDA line during the low phase of the ACK/NACK-related clock period (period 9), so that the SDA line is stable low during the high phase of the ACK/NACK-related clock period. Setup and hold times must be taken into account.

When the SDA line remains high during the ACK/NACK-related clock period, this is interpreted as a NACK. There are several conditions that lead to the generation of a NACK:

1. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.
2. During the transfer, the receiver gets data or commands that it does not understand.
3. During the transfer, the receiver cannot receive any more data bytes.
4. A master-receiver is done reading data and indicates this to the slave through a NACK.

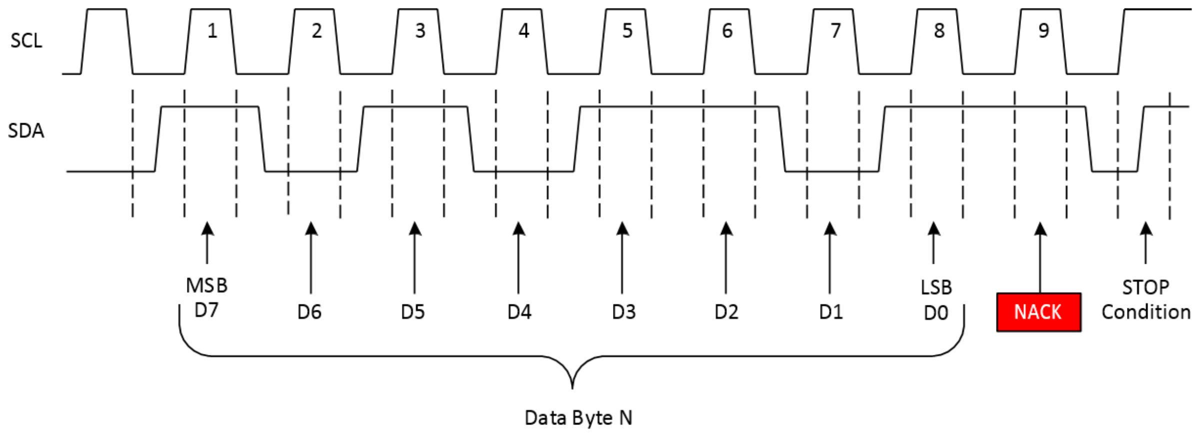

Figure A3. Example NACK Waveform

A2 I^{2}C Data

Data must be sent and received to or from the slave devices, but the way that this is accomplished is by reading or writing to or from registers in the slave device.

Registers are locations in the slave's memory which contain information, whether it be the configuration information, or some sampled data to send back to the master. The master must write information into these registers in order to instruct the slave device to perform a task.

While it is common to have registers in I^{2}C slaves, please note that not all slave devices will have registers. Some devices are simple and contain only 1 register, which may be written directly to by sending the register data immediately after the slave address, instead of addressing a register. An example of a single-register device would be an 8-bit I^{2}C switch, which is controlled via I^{2}C commands. Since it has 1 bit to enable or disable a channel, there is only 1 register needed, and the master merely writes the register data after the slave address, skipping the register number.

A2.1 Writing to a Slave On The I^{2}C Bus

To write on the I^{2}C bus, the master will send a start condition on the bus with the slave's address, as well as the last bit (the R/W bit) set to 0, which signifies a write. After the slave sends the acknowledge bit, the master will then send the register address of the register it wishes to write to. The slave will acknowledge again, letting the master know it is ready. After this, the master will start sending the register data to the slave, until the master has sent all the data it needs to (sometimes this is only a single byte), and the master will terminate the transmission with a STOP condition.

Figure A4 shows an example of writing a single byte to a slave register.

▨ Master Controls SDA Line

□ Slave Controls SDA Line

Write to One Register in a Device

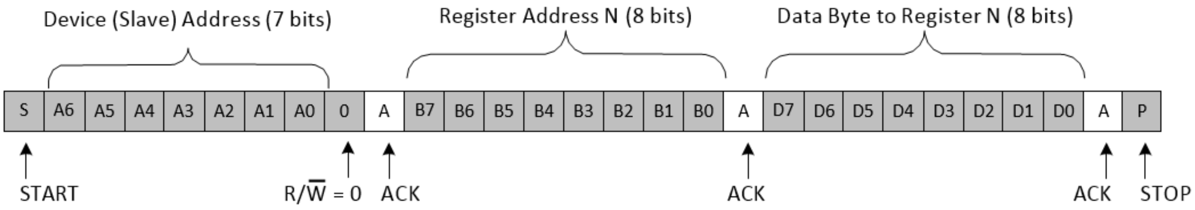
Device (Slave) Address (7 bits)
Register Address N (8 bits)
Data Byte to Register N (8 bits)
S
A6
A5
A4
A3
A2
A1
A0
0
A
B7
B6
B5
B4
B3
B2
B1
B0
A
D7
D6
D5
D4
D3
D2
D1
D0
A
P
START
R/W̄ = 0
ACK
ACK
ACK
STOP

Figure 4. Example I^{2}C Write to Slave Device's Register

A2.2 Reading *From a Slave On The I^{2}C Bus*

Reading from a slave is very similar to writing, but with some extra steps. In order to read from a slave, the master must first instruct the slave which register it wishes to read from. This is done by the master starting off the transmission in a similar fashion as the write, by sending the address with the R/W bit equal to 0 (signifying a write), followed by the register address it wishes to read from. Once the slave acknowledges this register address, the master will send a START condition again, followed by the slave address with the R/W bit set to 1 (signifying a read). This time, the slave will acknowledge the read request, and the master releases the SDA bus, but will continue supplying the clock to the slave. During this part of the transaction, the master will become the master-receiver, and the slave will become the slave-transmitter.

The master will continue sending out the clock pulses, but will release the SDA line, so that the slave can transmit data. At the end of every byte of data, the master will send an ACK to the slave, letting the slave know that it is ready for more data. Once the master has received the number of bytes it is expecting, it will send a NACK, signaling to the slave to halt communications and release the bus. The master will follow this up with a STOP condition.

Figure A5 shows an example of reading a single byte from a slave register.

▨ Master Controls SDA Line

□ Slave Controls SDA Line

Read Fro m One Register in a Device

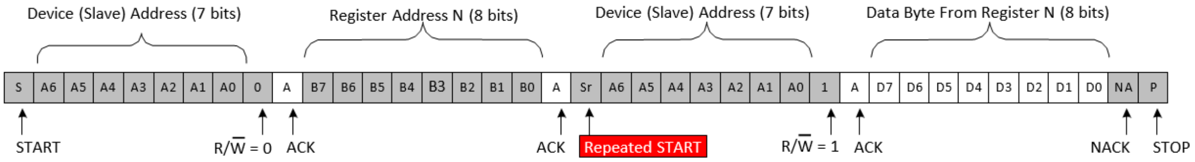
Device (Slave) Address (7 bits)
Register Address N (8 bits)
Device (Slave) Address (7 bits)
Data Byte From Register N (8 bits)
S
A6
A5
A4
A3
A2
A1
A0
0
A
B7
B6
B5
B4
B3
B2
B1
B0
A
Sr
A6
A5
A4
A3
A2
A1
A0
1
A
D7
D6
D5
D4
D3
D2
D1
D0
NA
P
START
R/W = 0
ACK
ACK
Repeated START
R/W = 1
ACK
NACK
STOP

Figure A5. Example I^{2}C Read from Slave Device's Register