Product Document

Published by
ams OSRAM Group

Application Note

AS7265x Multi Spectral Chipset

Design Considerations

Content Guide

1 General Description 3
2 AS7265x Multispectral Chipset 3
3 Hardware Design Considerations 4
3.1 UART Interface 7
3.2 I²C Interface 7
3.3 Light Source Selection 8
3.4 Other Connections 8
3.5 PCB Layout Considerations 9
3.6 Optic Considerations 10
4 Software Design Considerations 10
4.1 Features and Register Structure 10
4.2 I²C Virtual Register Read 11
4.3 I²C Virtual Register Write 13
5 Contact Information 15
6 Copyrights & Disclaimer 16
7 Revision Information 17

1 General Description

This Application Note briefly describes system level design considerations with ams AS7265x Multispectral Chipset solution.

2 AS7265x Multispectral Chipset

AS7265x Multispectral Chipset consists of AS72651, AS72652, and AS72653 devices and each device has 6 optical filters so that AS7265x Multispectral Chipset have total 18 channels for spectral identification from 400nm to 1000nm with FWHM of 20nm.

AS72651 is the main device with the smart interface to a microcontroller. The microcontroller gets the sensors data including AS72652 and AS72653 through AS72651 AT commands or I^{2}C registers. AS72651 requires a flash memory to work with and the flash memory contains AS7265x Multispectral Chipset firmware[1].

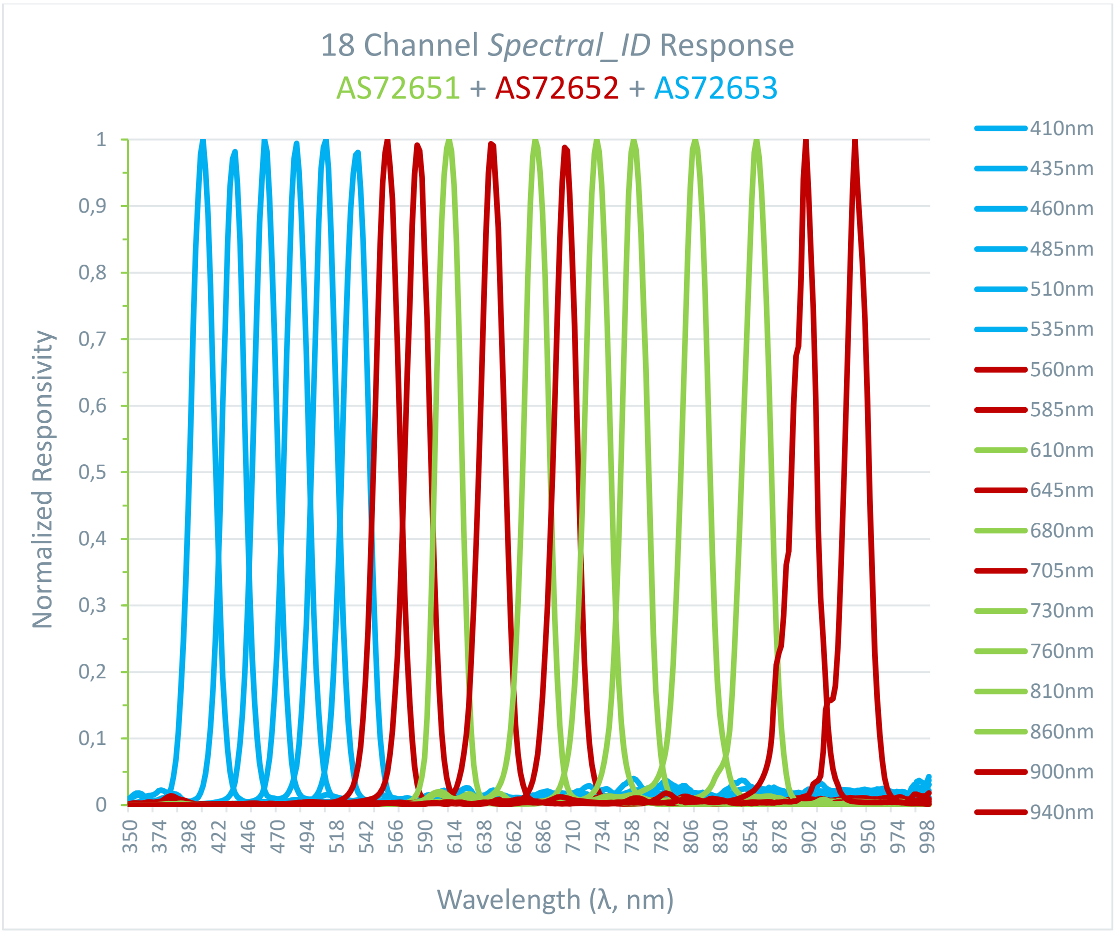
Each AS72651, AS72652, or AS72653 is packaged in 20-pin LGA package.

Figure 1. AS7265x Multispectral Chipset Optical Filters

Device	Channel	Filter Type	Center λ (nm)	FWHM (nm)
AS72653	A	Gaussian/BP	410	20
AS72653	B	Gaussian/BP	435	20
AS72653	C	Gaussian/BP	460	20
AS72653	D	Gaussian/BP	485	20
AS72653	E	Gaussian/BP	510	20
AS72653	F	Gaussian/BP	535	20
AS72652	G	Gaussian/BP	560	20
AS72652	H	Gaussian/BP	585	20
AS72651	R	Gaussian/BP	610	20
AS72652	I	Gaussian/BP	645	20
AS72651	S	Gaussian/BP	680	20
AS72652	J	Gaussian/BP	705	20
AS72651	T	Gaussian/BP	730	20
AS72651	U	Gaussian/BP	760	20
AS72651	V	Gaussian/BP	810	20
AS72651	W	Gaussian/BP	860	20
AS72652	K	Gaussian/BP	900	20
AS72652	L	Gaussian/BP	940	20

[1] See application note „AS72xx External Flash program and update“

Figure 2. Typical Spectral Responsivity



3 Hardware Design Considerations

Depending on the used firmware releases, there exists for AS7265x schematic two alternative versions, named generation 1 and 2. Compared with the older generation 1, in AS7265x new generation 2, the INT and SLV_RST1 pin are swapped on default. Note, all components for generation 1 and 2 in firmware, software and hardware are not compatible.

The following figures show the schematics for generation 1 and 2.

Figure 4. Typical Schematic AS7265x (Version FW generation 1)

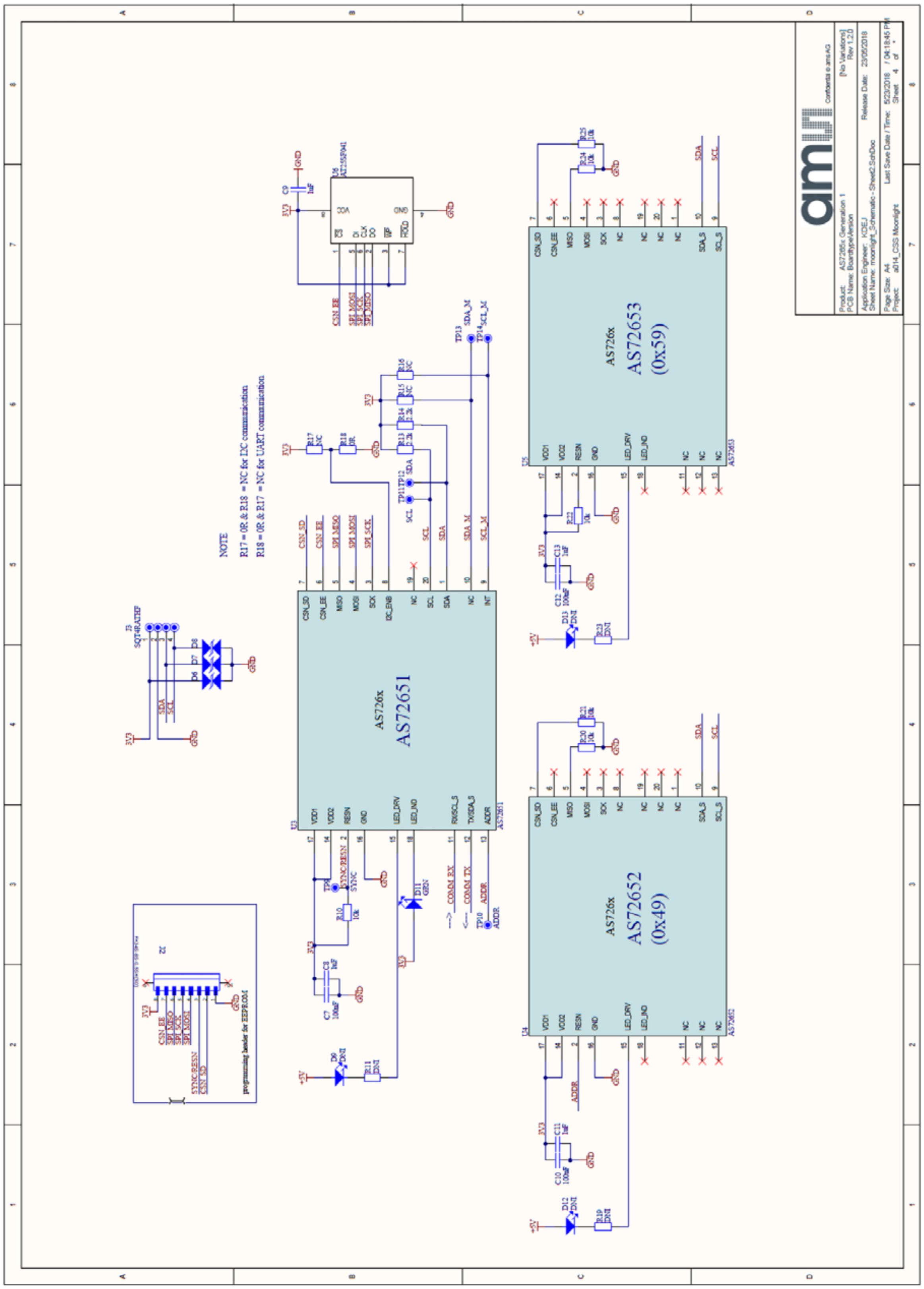
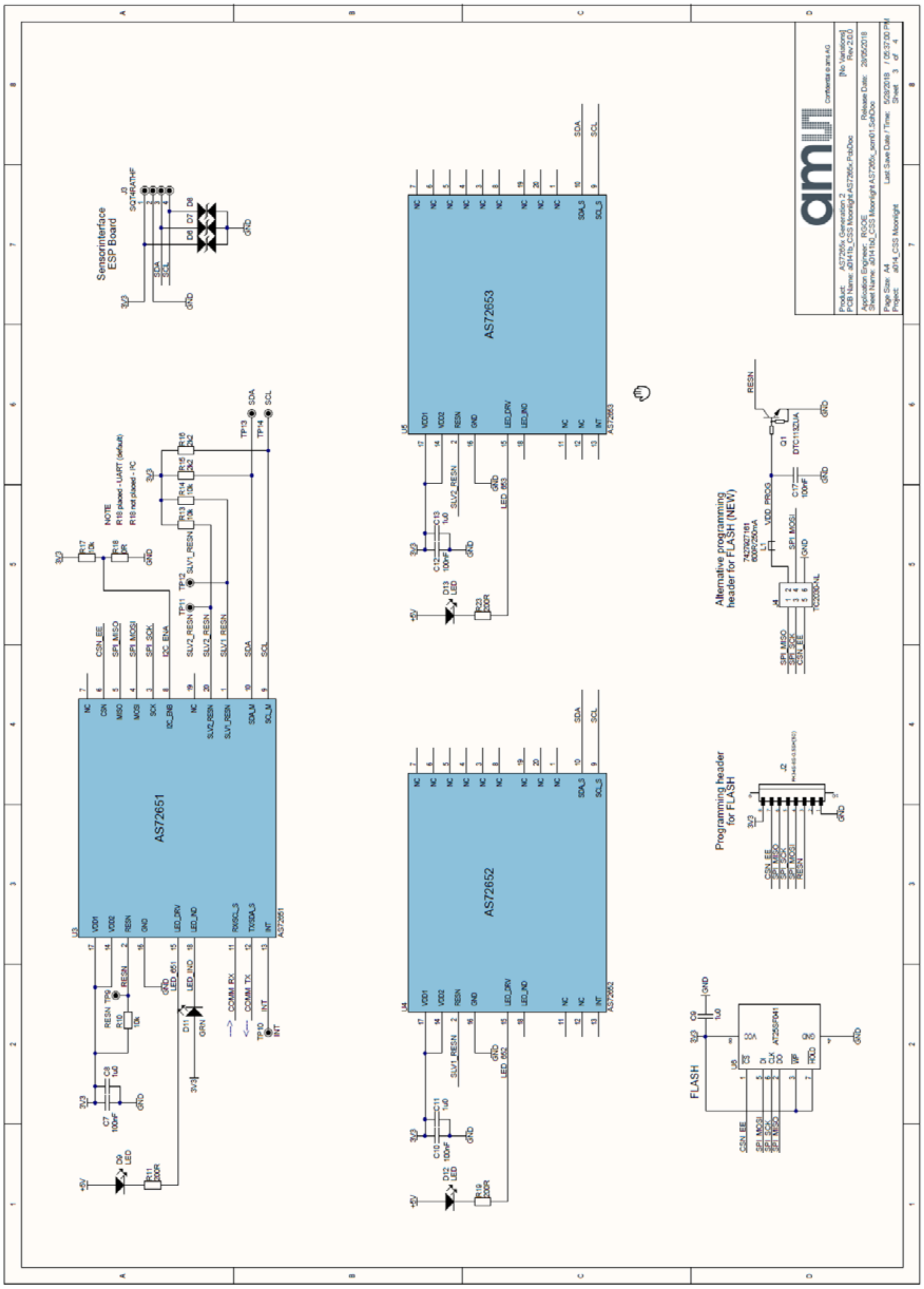


Figure 5. Typical Schematic AS7265x (Version FW generation 2)


AS72651
AS72652
AS72653
Sensorinterface ESP Board
Alternative programming header for FLASH (NEW)
Programming header for FLASH
FLASH
AT25SF041
DTC1132UA
NOTE
R18 placed - UART (default)
R18 not placed - I²C
Product: AS7265x Generation 2
PCB Name: a0141b_CSS Moonlight AS7265x.PcbDoc
Application Engineer: RGOE
Sheet Name: a0141b0_CSS Moonlight AS7265x_scm01.SchDoc
Page Size: A4
Project: a014_CSS Moonlight
Release Date: 28/05/2018
Rev 2.0.0
Last Save Date / Time: 5/28/2018 / 05:37:00 PM
Sheet 3 of 4
Confidential © ams AG

3.1 UART Interface

AS72651 has an UART interface to communicate to the controller. AT commands can be used for data acquisition, sensors configuration, and LED drivers control. Please refer to AS72651 data sheet for complete AT commands.

Pin11 of AS72651 is the RX of UART, which AS72651 receives the information from the controller. Pin12 of AS72651 is the TX of UART, which AS72651 transmits the information to the controller. Any Windows terminal application with baud rate 115200, 8 data bit, 1 stop bit, and none parity can be used for AT commands.

Since pin11 and pin12 of AS72651 are also shared with I^{2}C interface, the pin8, I^{2}C_ENB, has to be pulled down for UART interface configuration.

3.2 I^{2}C Interface

AS72651 has both I^{2}C master and I^{2}C slave interface. Both support I^{2}C fast mode (400 KHz) and standard mode (100 KHz).

AS72651 I^{2}C slave interface is used for communication to the controller. The pin11, SCL_S, is assigned to the I^{2}C bus clock and the pin12, SDA_S, is for the bus data. The pin8, I^{2}C_ENB, has to be pulled HIGH.

AS72651 I^{2}C master interface is used for controlling AS72652 and AS72653. The pin20, SCL, is the I^{2}C bus clock and the pin1, SDA, is for the I^{2}C bus data. The communication between AS72651 and AS72652/AS72653 is managed by the firmware.

According to I^{2}C specification, both SCL and SDA are open drain and need to be connected to a positive supply voltage via a pull-up resistor. The pull-up resistors, R13/R14 in the typical schematic, pull the line high when it is not driven low by the open drain interface. The maximum value of the pull-up resistor is limited by the bus capacitance, C_b, and the rise time, t_r, as below.

$$R_{P(max)} = \frac{t_r}{(0.8473 * C_b)}$$

The bus capacitance is the total capacitance of wire, connections, and pins. I^{2}C Bus specifies the maximum rise time is 300ns.

On the other hand, the minimum value of the pull-up resistor depends on the device logical specifications and allows V_{OL} level to be read as a valide logical low.

$$R_{P(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}}$$

For the AS7265x Multispectral Chipset application with 3.3V supply voltage, 0.4V maximum V_{OL}, and the specified minimum sink current of 3mA for standard mode (100 KHz) or fast mode (400 KHz), the minimum pull-up resistor value is 966.7Ω.

Then the decision of the pull-up resistor value would be based on the rise time, the total bus capacitance, and the power budget. A smaller resistor may get short rise time but has higher power consumption.

Please note, the typical schematic here is configured as UART interface by default so there are no pull-up resistors on AS72651 I²C slave interface. If I²C slave interface is needed, please add the pull-up resistors on either the controller side or on AS72651 I²C interface.

3.3 Light Source Selection

Figure 5. Generic Application

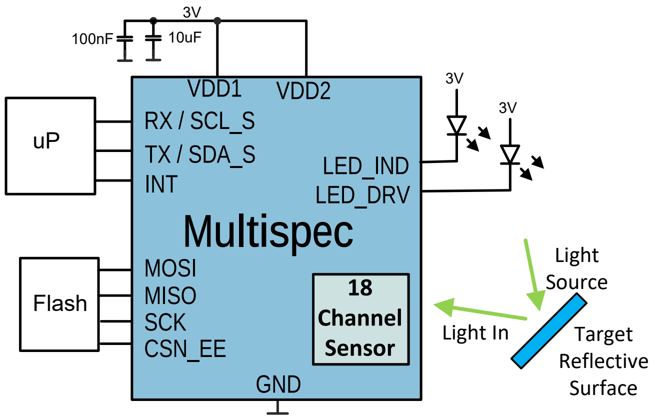


Figure 5 shows AS7265x Multispectral Chipset generic application. AS7265x chipset produce the sensor output data based on the received reflection light rays from the target. The light source selection would be dependent on the spectral responsivity of reflected light and characteristics of the target. For example, if the target is expected to absorb 610nm light in visible range and the application needs to distinguish the target from others, a broadband white LED might be used as the light source for AS7265x and AS7265x 610nm channel should be checked. Various applications may require different light sources.

3.4 Other Connections

The AS72651 device needs a flash memory to store the firmware and the data. The flash memory should be at least 2-Mbits operating at SPI mode 0 with byte write supported. With CSN_SD[i] pin and RESN pin, AS72651 supports various flash memory programming methodologies. Please refer to Application Note “AS72xx Flash program and update”.

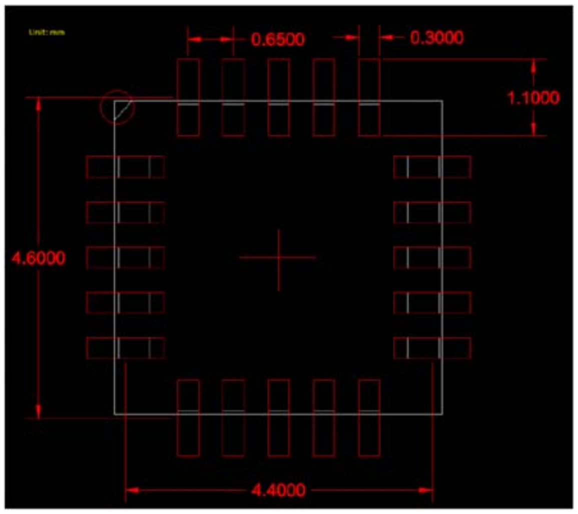
The LED, D11 in the typical schematic, is recommended for debugging purpose. During AS72651 power up, D11 should be on for a short time and off. If D11 is blinking, it indicates there is an issue with accessing the flash memory content.

The AS72651/AS72652/AS72653 devices require a 3.3V supply on both VDD1 and VDD2 pins associated with the decoupling capacitors, C7/C8, C10/C11, and C12/C13 in the schematic. Each LED_DRV pin on AS72651/AS72652/AS72653 can drive external LED sources with various amount of sink currents. The LED current can be configured as 12.5mA, 25mA, 50mA and 100mA.

3.5 PCB Layout Considerations

AS72651/AS72652/AS72653 has same 20-pin LGA package sharing same PCB footprint.

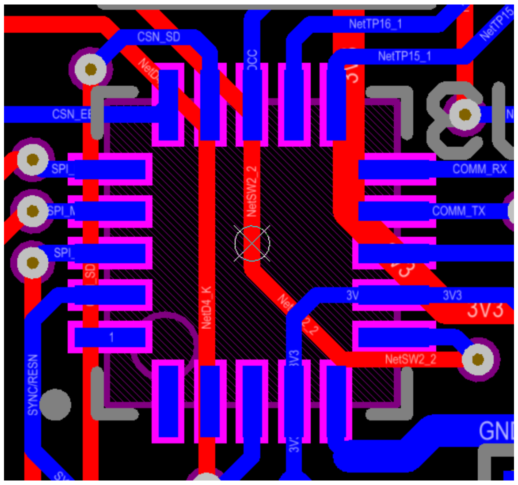
Figure 6. AS72651/AS72652/AS72653 PCB Footprint Recommendation



The schematic symbol and PCB layout footprint can also be provided in Altium Design format. Please contact ams support team to get the library file.

The PCB layout for AS72651/AS72652/AS72653 devices is simple. The first recommendation is to place the decoupling capacitors closed to VDD pins of AS72651/AS72652/AS72653 devices. The second recommendation is to avoid to put any via underneath the device. Please refer to the Figure below.

Figure 7. Sample Layout of AS72651/AS72652/AS72653 Devices



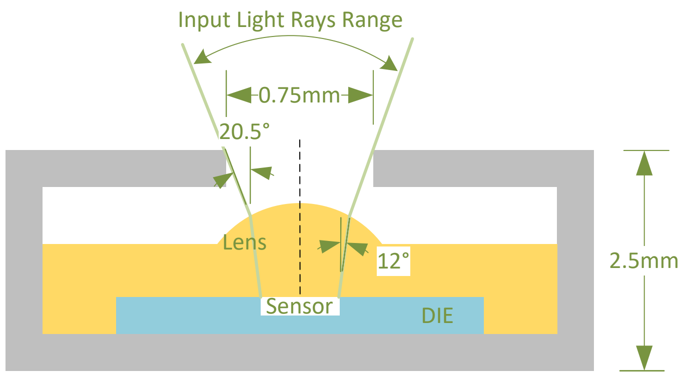
For the system level layout consideration, the generic PCB layout rules for digital designs should apply. In general, the wiring must be chosen so that crosstalk and interference to/from the bus lines is minimized. The I²C bus specification also recommends that place VDD and/or GND between SDL and SDA if the traces are longer than 10cm.

The length of I²C bus depends on the load of the bus and the speed you run at. The I²C bus specification defines the maximum capacitance of the bus is 400pF. This bus capacitance limit is specified to limit rise time reductions and allow operating at the rated frequency. In general, with lower frequency and/or lower capacitance of the bus, you can have longer bus length.

For most of I²C bus designs, the capacitance limit should be not the problem at all. If you design involves some unusual conditions, the specification has several strategies to cope with excess bus capacitance. For example, higher drive outputs, bus buffers, switched pull-up circuit etc. Please refer to the specification Section 7.2.

3.6 Optic Considerations

Figure 8. Aperture



Each AS7265x device has an open aperture on the surface. The diameter is 0.75mm and the package field of view is ±20.5°. The light rays in the range as shown in above figure would arrive at the sensor.

Since AS7265x Multispectral Chipset consists of three AS7265x devices, an external optical device might be needed so incident rays to each device is same.

As an open-aperture device, precautions must be taken to avoid particulate or solvent contamination as a result of any manufacturing processes, including pick and place, reflow, cleaning, integration assembly and/or testing.

4 Software Design Considerations

In most of system designs, AS72651 is controlled by a microcontroller. With the UART interface, the controller could configure the devices and get the sensors data through some AT commands. The software of microcontroller design would be simple.

The following sections would focus on I²C interface and the software of microcontroller design should satisfy both I²C specification and AS7265x Multispectral Chipset register structure.

4.1 Features and Register Structure

AS72651 supports I²C both standard mode and fast mode. The addressing mode is 7+1-bit so when the controller send a read command to AS72651, the salve address plus R/W bit should be 0x93 and when sending a write command, it should be 0x92. Both read and write are single byte process.

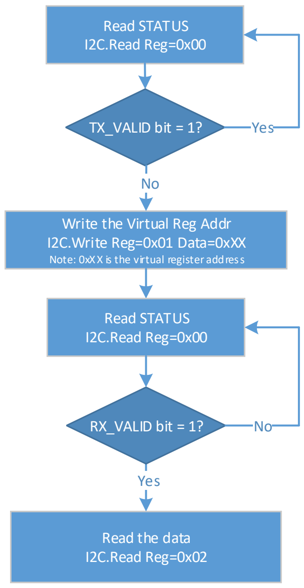
AS72651 does not support the slave clock stretching mode.

AS72651 has only three hardware based registers, STATUS (0x00), WRITE (0x01), and READ (0x02). The rest are implemented as virtual registers in the firmware. All virtual registers are accessed through WRITE and/or READ registers. Please refer to the data sheets for complete set of virtual registers.

4.2 I^{2}C Virtual Register Read

To read an I^{2}C virtual register, please follow the flow chart below.

Figure 9. Flow Chart for Virtual Register Read



To poll the STATUS register, the controller should write the STATUS address then following a read command to get the value of the STATUS register. The Figure 3 shows the format of the command for polling the STATUS register.

Figure 10. Command for Polling the STATUS Register

Start	0x92	STATUS	Ack	Repeat Start	0x93	Data	Nack	Stop

To write the virtual register address, please program WRITE register with the virtual register address as the following format.

Figure 11. Command for Writing the Virtual Register Address for Reading

Start	0x92	WRITE	Ack	Virtual Reg Addr	Ack	Stop

Finally below is the reading command to get the data.

Figure 12. Command for Reading the READ register

Start	0x92	READ	Ack	Repeat Start	0x93	Data	Nack	Stop

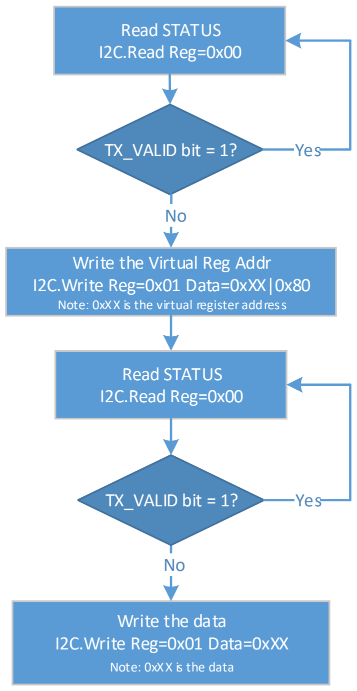
Figure 13. Sample Code of Reading a Virtual Register

```
#define I2C_AS72XX_SLAVE_STATUS_REG    0x00
#define I2C_AS72XX_SLAVE_WRITE_REG     0x01
#define I2C_AS72XX_SLAVE_READ_REG      0x02
#define I2C_AS72XX_SLAVE_TX_VALID      0x02
#define I2C_AS72XX_SLAVE_RX_VALID      0x01

uint8_t i2cm_AS72xx_read(uint8_t virtualReg)
{
      volatile uint8_t status, d ;

      while (1)
      {
            // Read slave I2C status to see if we can write the reg address.
            status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;

            if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
                  // No inbound TX pending at slave.  Okay to write now.
                  break ;
      }
      // Send the virtual register address
      i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, virtualReg) ;

      while (1)
      {
            // Read the slave I2C status to see if our read data is available.
            status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;

            if ((status & I2C_AS72XX_SLAVE_RX_VALID) != 0)
                  // Read data is ready for us.
                  break ;
      }
      // Read the data to complete the operation.
      d = i2cm_read(I2C_AS72XX_SLAVE_READ_REG) ;
      return d ;
}
```

4.3 I²C Virtual Register Write

Writing to a virtual register is similar to the read.

Figure 14. Flow Chart for Virtual Register Write



Please refer to the previous section for polling the STATUS register.

Writing the virtual register address for writing is not same as the one for reading. The MSB of the virtual register address has to be set to 1 for writing.

Figure 15. Command for Writing the Virtual Register Address for Writing

Start	0x92	WRITE	Ack	Virtual Reg Addr | 0x80	Ack	Stop

Simple command for writing the data as below.

Figure 16. Command for Writing the Data

Start	0x92	WRITE	Ack	Data	Ack	Stop

Figure 17. Sample Code of Writing a Virtual Register

```
void i2cm_AS72xx_write(uint8_t virtualReg, uint8_t d)
{
      volatile uint8_t    status;

      while (1)
      {
            // Read slave I2C status to see if we can write the reg address.
            status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG);

            if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
                  // No inbound TX pending at slave.  Okay to write now.
                  break ;
      }
      // Send the virtual register address
      // (setting bit 7 to indicate a pending write).
      i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, (virtualReg | 0x80)) ;

      while (1)
      {
            // Read the slave I2C status to see if we can write the data byte.
            status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;

            if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
                  // No inbound TX pending at slave.  Okay to write data now.
                  break ;
      }
      // Send the data to complete the operation.
      i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, d) ;
}
```

5 Contact Information

Buy our products or get free samples online at:

www.ams.com/ICdirect

Technical Support is available at:

www.ams.com/Technical-Support

Provide feedback about this document at:

www.ams.com/Document-Feedback

For further information and requests, e-mail us at:

ams_sales@ams.com

For sales offices, distributors and representatives, please visit:

www.ams.com/contact

Headquarters

ams AG
Tobelbaderstrasse 30
8141 Premstaetten
Austria, Europe

Tel: +43 (0) 3136 500 0
Website: www.ams.com

6 Copyrights & Disclaimer

7 Revision Information

Changes from previous version to current revision 1-03 (2017-Mar-15)	Page
Updated Figure 2	4
Added Light Source Selection and Optic Considerations	7, 9
Updated Moonlight with AS7265x Multispectral Chipset	
New generation of AS7265x firmware and hardware and splitting in gen 1 and 2	

Note: Page numbers for the previous version may differ from page numbers in the current revision.
Correction of typographical errors is not explicitly mentioned.

[i] Depend on the AS7265x generation – see schematics, see chapter **Error! Reference source not found.**